BYTE / McGRAW-HILL

Construa seu próprio Microcomputador

STEVE CIARCIA

McGRAW-HILL

Construa o seu próprio microcomputador Z80

Construa o seu próprio microcomputador Z80

STEVE CIARCIA

Tradução
Edson Bonfim de Souza
Paulo Salgueiro R. Franco

Revisão Técnica
Arnaldo Milstein Mefano
Professor Assistente do Departamento de Eletrônica
da Faculdade de Engenharia – UERJ

MAKRON *Books* do Brasil Editora Ltda.
Editora McGraw-Hill Ltda.
São Paulo
Rua Tabapuã, 1105, Itaim-Bibi
CEP 04533
(011) 829-8604 e (011) 820-8528

Rio de Janeiro • Lisboa • Porto • Bogotá • Buenos Aires • Guatemala • Madrid • México • New York • Panamá • San Juan • Santiago

Auckland • Hamburg • Kuala Lumpur • London • Milan • Montreal • New Delhi • Paris • Singapore • Sydney • Tokyo • Toronto

Do original

Build Your Own Z80 Computer

Composição e Arte: Brasil Artes Gráficas Ltda.

Capa: Viviane Malhamé

CIP-Brasil. Catalogação-na-Publicação

Câmara Brasileira do Livro, SP

Ciarcia, Steve.
C495c Construa o seu próprio computador usando o MP-Z80 / Steve Ciarcia; tradução Edson Bonfim de Souza, Paulo Salgueiro R. Franco; revisão técnica Arnaldo Milstein Mefano. – São Paulo: McGraw-Hill do Brasil, 1984.

1. Computadores eletrônicos digitais – Manual para amadores. 2. Z-80 (Computador). I. Título.

83-1580 17. CDD-621.381958
18. -621.3819582

Índices para catálogo sistemático:

1. Computadores eletrônicos digitais: Manual para amadores: Engenharia 621.381958(17.) 621.3819582(18.)
2. Modelo Zilog Z-80: Computadores digitais: Construção: Engenharia eletrônica 621.381958(17.) 621.3819582(18.)
3. Z-80: Modelo Zilog: Computadores digitais: Construção: Engenharia eletrônica 621.381958(17.) 621.3819582(18.)

Para minha esposa Joyce,
Steve Sunderland, e Judy e Lloyd Kishinsky

SUMÁRIO

INTRODUÇÃO

Há alguns anos atrás, quando foram introduzidos os microprocessadores, os entusiastas do computador e engenheiros elétricos eram os mesmos: aqueles de nós que viveram somente para soldar, aquecidos em nossa glória. Agora, porém, os preços de sistemas completamente montados e embalados têm-se nivelado. Hoje qualquer um com interesse pode possuir e operar um computador. A compra de um computador hoje em dia é similar à compra de um aparelho de televisão e a classe dos entusiastas do computador tem aumentado muito.

Como qualquer movimento popular, a literatura disponível reflete o interesse da maioria dos seguidores.

De acordo com a popularização do computador, a ênfase técnica nos computadores de estante está afastada do projeto de *hardware*. Por outro lado, muitos livros sobre computador, com textos introdutórios do tipo COMO FUNCIONAM AS PORTAS LÓGICAS, tratam o microcomputador de forma por demais simplista, como se fosse um livro de receitas, muitas vezes omitindo os ingredientes principais. Geralmente, as únicas alternativas são os textos de engenharia ou os jornais especializados, nem sempre fáceis de se compreender.

Há alguns anos eu venho escrevendo uma coluna na revista BYTE, e a resposta dos leitores tem mostrado que ainda existe um grande interesse nos projetos de *hardware* e projetos Faça-você-mesmo. Ao mesmo tempo, eu fico apreensivo com a carência de material para esse pessoal. Muitas correspondências vêm de estudantes de escolas técnicas ou secundárias que leram todas as descrições e estudaram os diagramas de blocos, mas que desejam respostas práticas e exemplos de sistemas. Infelizmente, existem poucos livros que eu possa sugerir.

CONSTRUA O SEU PRÓPRIO COMPUTADOR USANDO O MP–Z80 é um livro escrito para indivíduos tecnicamente médios que estão interessados em saber como é um microcomputador por dentro. É para pessoas que já tenham um entendimento básico de eletrônica, e que desejam construir um computador em vez de comprar. Não é um livro de introdução à eletrônica, que inicia pela descrição das portas lógicas, nem é, por outro lado, um texto escrito somente para estudantes de engenharia. Servindo para educar o curioso, o objetivo deste livro é apresentar uma análise prática, passo-a-passo, da arquitetura de um computador digital, e os detalhes de construção de um completo e funcional microcomputador.

O computador a ser construído é chamado Processador de Aplicações Z80–PAZ. Ele está baseado no componente microprocessador Z80 da Zilog. Esse componente foi escolhido tomando-se por base sua eficácia e baixo custo, como os outros componentes do PAZ. Para ajudar o entusiasta caseiro, e para aqueles experimentadores que preferem começar um livro pelo final, eu listei no Apêndice A uma companhia que fornece os EPROMs programados (ERASABLE–PROGRAMMABLE READ–ONLY MEMORY).

Eu estruturei o livro como uma seqüência lógica de marco de construção entremeada por discussões práticas da teoria de operação. Meu propósito é duplo: ajudar um construtor potencial a ganhar confiança, e tornar o material mais apetitoso através de exemplos concretos.

Este é basicamente um manual de construção; subsídios consideráveis são dados para os "porquês" e "comos" do projeto do computador. Ao leitor são expostos vários assuntos, incluindo: as arquiteturas internas de microprocessadores selecionados, mapeamento de memória, interface de entrada/saída, fontes de alimentação, comunicação com periférico, e programação. Todos os discursos tentam tornar o leitor inteirado dos efeitos de cada componente individual no sistema total. Embora eu tenha documentado os detalhes específicos do computador PAZ, é minha intenção (e a premissa do livro) que o leitor seja capaz de configurar um computador. O PAZ é uma ferramenta experimental que pode ser expandida para ir de encontro a uma variedade de aplicações.

O PAZ é construído como uma série de subsistemas que podem ser testados e exercitados independentemente. O primeiro item a ser construído é a fonte de alimentação. Esta é uma boa maneira de testar habilidade e prover de imediato uma realimentação para uma construção bem sucedida. As três fontes de tensão são protegidas em temperatura e sobretensão e têm uma corrente adequada para a expansão do sistema PAZ.

Depois, o leitor aprende por que o Z80 foi escolhido para o PAZ e considerações de arquitetura que afetam a seleção de componentes em outros subsistemas. Um capítulo inteiro é devotado ao integrado Z80. Cada sinal de controle é explicado em detalhe e cada instrução é cuidadosamente documentada.

A construção de *hardware* prossegue em estágios com testes intermediários para assegurar o sucesso da montagem. Os elementos básicos do computador são montados primeiro e então testados. O leitor seleciona quais periféricos estão para ser adicionados. O livro contém seções de construção de um mostrador hexadecimal, teclado, programador de EPROM, interface serial RS-232C, sistema de gravação em cassete, e um terminal de vídeo e mais um capítulo relativo à interface do PAZ com sinais analógicos. Eu forneço circuitos específicos que podem converter o PAZ em um sintetizador de voz digital ou um sistema de aquisição de dados.

Um monitor de *software* especial de 1K (1024 BYTES) coordena as atividades básicas do computador e dos periféricos. O *software* é explicado através de fluxogramas e listagens. Com este monitor o PAZ pode funcionar como um terminal de computador, um controlador, ou um sistema de desenvolvimento de *software*.

Construa o seu próprio Computador usando o MP-Z80 é um livro para pessoas de *hardware*. Da fonte ao processador central, este livro é escrito para pessoas que querem entender o que elas constroem.

Steve Ciarcia
Maio, 1981

N.T.: Por serem algumas figuras cópias de manual, certos termos e até mesmo figuras inteiras não estão traduzidos, pois em português não teriam significado.

CAPÍTULO 1

FONTE DE ALIMENTAÇÃO

Não há mistério nenhum em construir-se um cartão com um processador central, memória e algumas entradas e saídas, e chamá-lo de computador. Porém, a partir do momento em que você liga a chave de alimentação do computador, o sistema passa a ser completamente dependente da operação adequada da sua fonte de alimentação. Um livro que se preocupe com a construção de um computador, desde o princípio, seria completamente inadequado sem uma descrição de como construir uma fonte de alimentação adequada.

Muito se tem escrito sobre fontes de alimentação e corrente contínua (CC), conversores de CC para CC e CA para CC, reguladores de derivação e chaveamento, transformadores de tensão constante, e assim por diante. Não é minha intenção fazer uma fonte de alimentação que satisfaça a todas as aplicações. Em vez disto, esboçarei o projeto da fonte de alimentação CC específica que usaremos para alimentar o Processador de Aplicações Z80 (PAZ).

Em grandes computadores, as fontes CC fornecem enormes quantidades de potência para suprir milhares de circuitos lógicos; por necessidade, os fabricantes escolhem os métodos mais eficientes de conversão de potência. Estes métodos detalhados seriam custosos e difíceis de se construir em um protótipo. Felizmente, a potência necessária para o PAZ é bem menor do que aquela para os grandes computadores; portanto podemos aproveitar as vantagens dos métodos dos modelos estabilizados e incorporarmos os últimos avanços da tecnologia dos reguladores. A figura 1.1 é um diagrama de blocos da fonte de alimentação para o PAZ.

Cada uma das três fontes CC necessárias para alimentação do PAZ consiste em três módulos básicos: uma seção de transformação para reduzir a tensão de 120 VCA da linha para uma tensão mais baixa usada pelo computador; um retificador com filtro de entrada para converter CA em CC com baixa ondulação; e um regulador que estabiliza a saída em um nível de tensão fixado. O circuito de proteção para sobretensão será discutido separadamente.

As especificações do transformador e do filtro de entrada são quase sempre negligenciadas pelos hobistas que passam por cima das conseqüências de um filtro pobremente projetado. Isto é causado, em parte, pela abundância de informações técnicas circuladas pelos fabricantes de semicondutores, exaltando as virtudes de seus circuitos reguladores. Pessoas desavisadas poderão concluir, a partir destes folhetos de publicidade, que a seção de regulação de uma fonte de alimentação é a única parte que merece consideração; realmente, os avanços nos projetos de reguladores e o advento dos reguladores de três terminais e alta potência têm reduzido a necessidade da aplicação de dispositivos analógicos. No passado componentes adicionais e consideráveis cálculos eram necessários para se produzir um regulador de tensão adequado; hoje, entretanto, a maioria destes reguladores pode ser acomodada em um único e compacto dispositivo. Porém a seção do filtro de entrada não deve ser desprezada e ainda requer consideração e importância para cada aplicação.

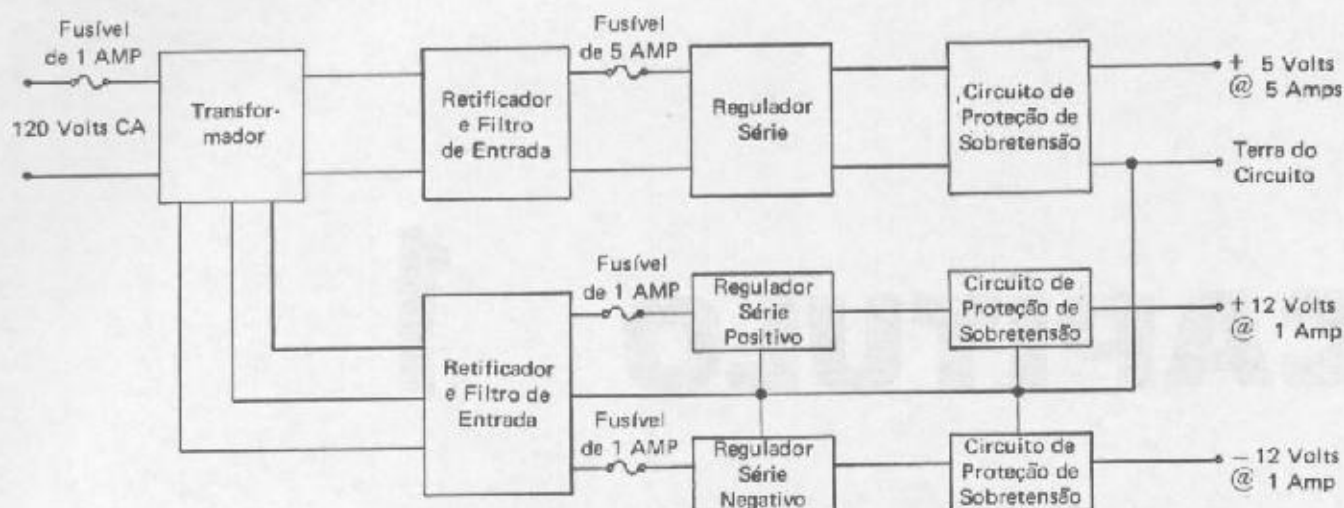



Figura 1.1 Um diagrama de bloco da fonte de alimentação básica para o processador de aplicações Z80 (PAZ)

Existem três fontes de tensão necessárias para a operação do PAZ. Cada fonte possui um filtro de entrada. Devido a fonte de + 5V ser a mais importante ela receberá maior atenção. Para o propósito desta discussão, dividiremos a fonte em duas seções: transformador e filtro de entrada, e regulador de saída.

A figura 1.2 mostra um diagrama de bloco de um filtro de entrada básico. Em sua forma mais simples, ele é constituído de três componentes que funcionam da seguinte forma:

- Um transformador que isola a fonte de alimentação propriamente dita da linha de alimentação CA, e reduz a entrada de 120 VCA para uma tensão CA mais baixa utilizável.
- Um retificador em ponte que converte CA em CC de onda completa e satisfaz a corrente de carga necessária ao capacitor de filtro.
- Um capacitor de filtro que mantém um nível de tensão suficiente, entre os ciclos de carga, para satisfazer as limitações de tensão da entrada do regulador.

PROJETANDO UM FILTRO DE ENTRADA

Você pode pensar que a especificação do transformador seja a primeira consideração quando se projeta uma fonte de alimentação. Sim e não. A aproximação da tensão de saída pode ser determinada por regras diversas, mas os valores exatos são deduzidos somente pela análise feita a partir da tensão de saída desejada. Na prática, a diferença entre uma expectativa razoável e uma análise laboriosa seria importante somente para uma pessoa capaz de construir seu próprio transformador. Na maioria dos casos teremos de utilizar transformadores com saídas padronizadas. Por esta razão, minha aproximação está mais para aspectos práticos de projetos de fonte de alimentação do que para detalhes minuciosos de engenharia que não têm realmente propósito no nosso caso. Para uma onda senoidal de 120 VCA RMS aplicada no primário do transformador, veremos na figura 1.2 a ilustração das formas de onda em pontos selecionados através da seção de filtro. A foto 1.1 mostra que a tensão de 120 VCA é na realidade de 340 V pico a pico; portanto deve-se tomar cuidado na isolação e montagem dos componentes.

O secundário terá uma onda senoidal similar, reduzida em tensão. Esta, então, é aplicada em uma ponte retificadora de onda completa, e a forma de onda aparecerá como na foto 1.2. Como o comportamento de componentes eletrônicos reais diferem de seus modelos matemáticos, devemos estar prevenidos para determinadas peculiaridades. Você notará um leve achatamento entre picos. Diodos de silícios possuem *threshold* (limiar de condução) característicos e, de fato, possuem uma queda de tensão de aproximadamente 1V em cada diodo. Esta queda de tensão torna-se significante em pontes de onda completa e, como ilustrado nas figuras 1.3a, 1.3b e foto 1.2, pode ser acumulada com a colocação de diodos em série. Os 2V perdidos na ponte é uma importante consideração e merece reflexão nos cálculos.

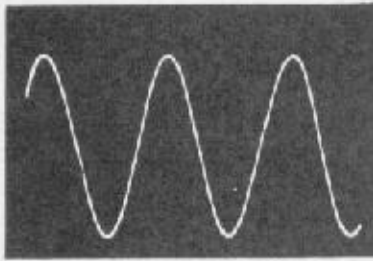



Foto 1.1 Forma de onda de entrada/saída (120 VCA RMS) de um transformador saturado

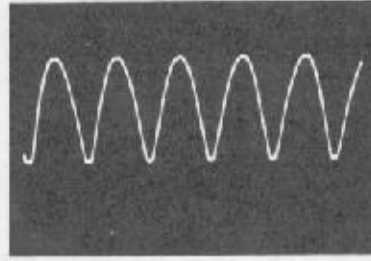



Foto 1.2 Forma de onda do retificador

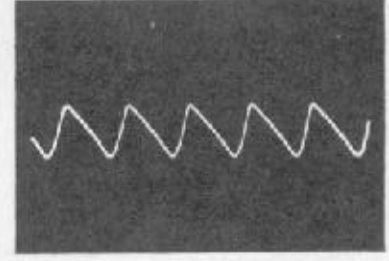



Foto 1.3 Forma de onda da ondulação em várias cargas

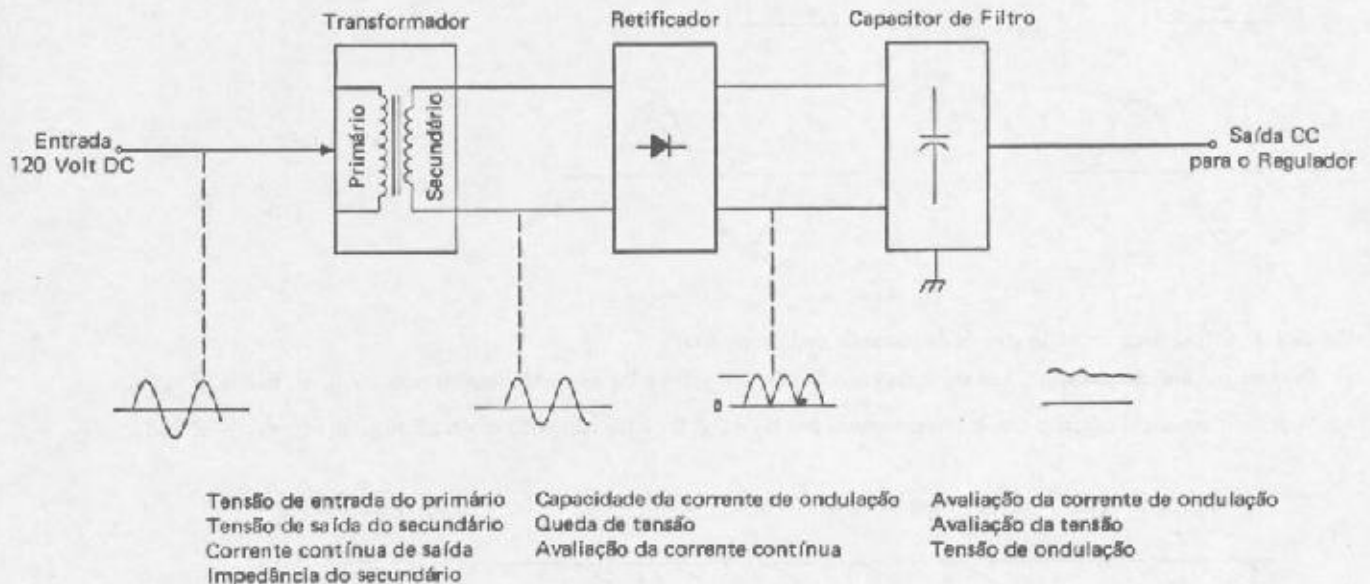



Figura 1.2 Diagrama em bloco de um filtro de entrada básico

O regulador de tensão requer um nível CC mínimo para manter constante a tensão de saída. Caso a tensão aplicada seja muito abaixo deste ponto, a estabilidade da saída estará severamente comprometida. Assim, um capacitor de filtro é usado para alisar as ondulações da onda senoidal retificada.

Quando os diodos estão conduzindo, o capacitor armazena energia suficiente para manter a tensão mínima necessária até o próximo ciclo de carga. A entrada do transformador está em 60 Hz, mas devido às características de retificação em onda completa, os ciclos de carga ocorrem em 120 Hz. A carga no capacitor leva um ciclo de 8,3 ms e, como o regulador puxa potência deste para satisfazer a carga demandada, este deve continuar provendo pelo menos a tensão de entrada mínima requerida pelo regulador até o próximo ciclo de carga, 8,3 ms mais tarde. Este fenômeno periódico carga/descarga está na foto 1.3. O tamanho da flutuação de tensão entre dois picos do ciclo é chamado de *ripple* (ondulação). A maior extensão da forma de onda incluindo o *ripple* é chamada de tensão de pico. Ambas são importantes lembrar e estão demonstradas na figura 1.4.

De posse de um entendimento básico dos componentes, podemos continuar com o nosso caso: uma fonte de alimentação de 5 V, 5 A. Por razões que discutiremos mais tarde, o regulador de 5 V desta fonte necessitará no mínimo de 8,5 V para a sua operação adequada. Isso significa que qualquer que seja a grandeza de V_{pico} e V_{ripple}, o nível final V_c não deve estar abaixo de 8,5 V ou o regulador não funcionará. Dando-nos alguma folga, faremos $V_c = 10$ V. Indo muito acima de 10 V, ainda que satisfaça o critério de entrada, poderá aumentar a dissipação de potência e provavelmente destruir o regulador. Existe uma resposta para este círculo vicioso e esta deve ser conservativa. A experiência mostra que um pouco de garantia vale a pena.

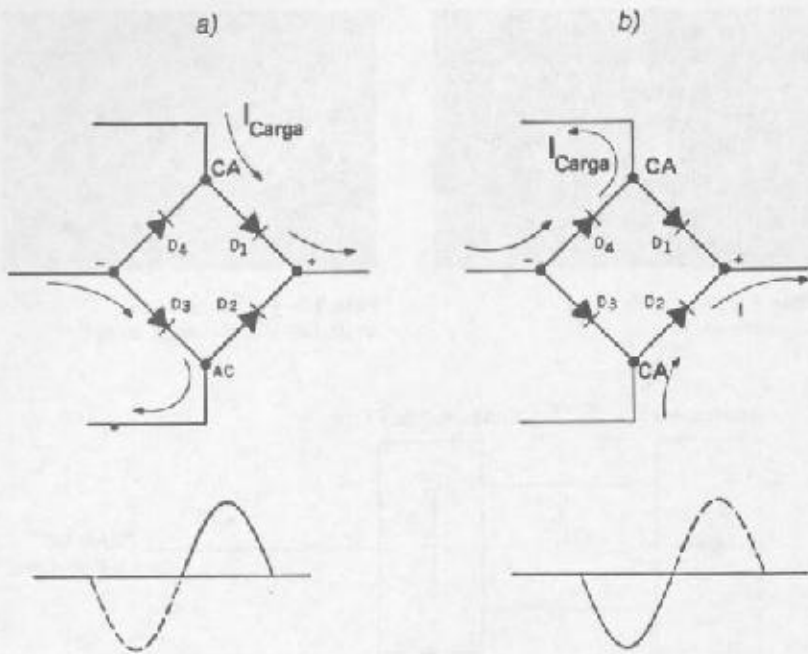



Figura 1.3 A direção da corrente através da ponte de onda completa.

a) Durante o semiciclo positivo CA, a corrente passa por D_1 e D_3; D_2 e D_4 não estão conduzindo. $V_{D1} + V_{D3} \approx 2$ *volts*.

b) Durante o semiciclo negativo CA, a corrente passa por D_2 e D_4; D_1 e D_3 não estão conduzindo. $V_{D2} + V_{D4} \approx 2$ *volts*.

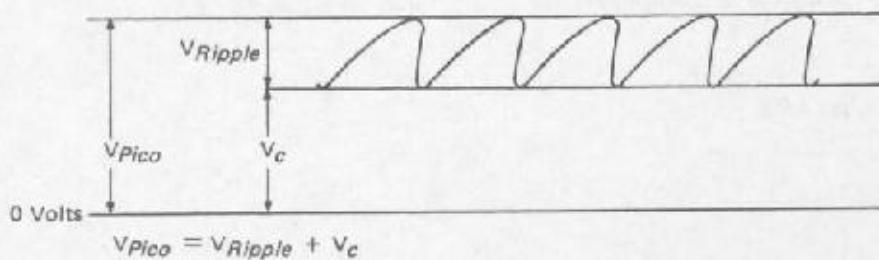



Figura 1.4 Tensão de saída como uma combinação de uma determinada tensão em estado estável (V_C) mais uma tensão de *ripple* (V_{ripple})

Agora que 10V é a meta, podemos selecionar apropriadamente os outros componentes para encontrá-la. A figura 1.5 é o circuito do filtro para a nossa fonte de 5V. R_s é a resistência do enrolamento do secundário de um transformador de 5 a 8A, esta resistência será em média cerca de 0,1 ohms. Os primeiros valores obtidos:

V_c = tensão mínima de entrada do regulador = 10V
I_{out} = carga do regulador = 5A
R_s = resistência do secundário do transformador = 0,1 ohms

V_{pico} pode ser qualquer tensão acima da entrada mínima para a qual o regulador esteja avaliado. Entretanto, isto aumentará a dissipação de potência do circuito. A regra que eu uso no projeto de pontes deste tipo é fazer V_{pico} ser aproximadamente 25% maior do que V_c. Neste caso o valor do capacitor será mantido dentro de limites razoáveis. A razão de V_c para ($V_{pico} - V_c$) está relacionada com o fator de *ripple* do capacitor de filtro.

$$Y_r = \frac{V_{pico} - V_c}{V_c} = \frac{12,5 - 10}{10} = 25\%$$

Um fator de *ripple* (ondulação) de 25% para 5 A ficará dentro das estimativas aceitáveis da corrente de *ripple* do capacitor e elimina a necessidade de se mergulhar nas especificações dos fabricantes de capacitores. Este fator de *ripple* é arbitrário, mas é melhor mantê-lo tão baixo quanto possível.

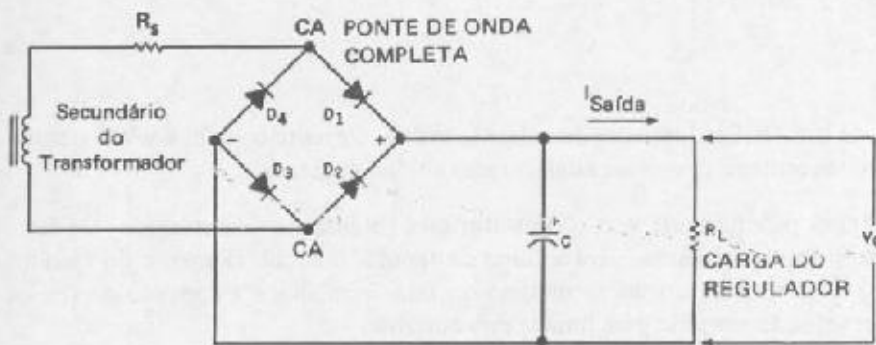



Figura 1.5 Circuito do filtro de entrada da fonte de alimentação de 5 V.

DIMENSIONANDO O CAPACITOR

Agora sabemos que o capacitor deve sustentar 10 V de uma entrada de 12,5 V de pico.

$$\left.\begin{aligned} V_{Pico} &= 12{,}5\ \text{V} \\ V_c &= 10\ \text{V} \\ V_{Ripple} &= 2{,}5\ \text{V} \end{aligned}\right\} \quad V_c = V_{Pico} - V_{Ripple}$$

A próxima consideração é a escolha de um capacitor adequada a esta meta. Uma outra regra de cálculo que elimina um extenso trabalho é:

$$C = \frac{dt}{dv}\, I$$

onde

C = valor do capacitor em farads = ?
I = corrente máxima do regulador = 5 A
dt = tempo de carga do capacitor = 8,3 ms (120 Hz)
dv = tensão de *ripple* admissível = 2,5 V

colocando em valores do nosso circuito,

$$C = \frac{(5)(8{,}3\times10^{-3})}{(2{,}5)} = 16{,}6\times10^{-3}\ \text{farads}$$

ou

$$C = 16{,}600\ \text{microfarads}\ (\mu F)$$

Normalmente capacitores eletrolíticos encontrados comercialmente têm uma tolerância de +50% e −20%. Para ficar num tamanho seguro e tornar fácil encontrar-se um componente básico, será melhor um valor de 20.000 μF. O acréscimo de 3400 μF reduz o *ripple* de outros 0,4 V e dá mais segurança. O único outro item a considerar do capacitor é a tensão de operação. Devido ao projeto ter especificado que V_{pico} é 12,5 V, esta deve ser uma medida satisfatória. Entretanto a experiência mostra que os transformadores acabam por fornecer tensões maiores do que as especificadas e que 12,5 V em 115 VCA alcança 13,6 V quando a tensão de linha sobe para 125 VCA. Uma tensão de 15 VCC para o capacitor pode parecer satisfatória, mas eu recomendo usar o valor superior mais próximo, ou seja, 20 VCC.

O capacitor então é de 20 μF para 20 VCC. O retificador pode ser uma ponte de onda completa monolítica, ou quatro diodos discretos. Repare que devido a ponte ser normalmente encapsulada, os quatro terminais estão especificados ao invés de se mostrarem as marcas de polaridades dos diodos individualmente. As designações para os quatro terminais são dois terminais de entrada CA e os terminais de saída + e −.

O RETIFICADOR

Existem três considerações na escolha de um retificador: valor da corrente inicial, corrente contínua e VPI (tensão de pico inverso). Essas escolhas não são inconseqüentes e devem ser consideradas cuidadosamente.

Quando uma fonte de alimentação é ligada pela primeira vez, o capacitor está totalmente descarregado. De fato, poderá parecer uma impedância de 0 ohm, instantaneamente, para a fonte de tensão. O único elemento do circuito que limita a corrente inicial é a resistência do enrolamento do secundário do transformador e a conexão do fio; os projetistas freqüentemente adicionam uma resistência em série para limitar esta corrente.

A corrente inicial neste circuito é

$$I_{in} = \frac{V_{pico}}{R_s} = \frac{12,5}{0,1} = 125 \text{ A}$$

e a constante de tempo do capacitor é

$$\tau \cong R_s \times C \cong (0,1)(20 \times 10^{-3}) = 2 \text{ ms}$$

Essa corrente não causará danos ao diodo se for menor do que a suportada pelo diodo e se

$$\tau < 8,3 \text{ ms}$$

Nós não podemos verificar esta corrente até que a ponte seja escolhida, mas os outros dois parâmetros podem ser definidos.

A ponte pode ser uma das duas abaixo

Motorola MDA 980-2: $I_{cont} = 12$ A, $I_{in} = 300$ A, PIV = 100 V
Motorola MDA 990-2: $I_{cont} = 27$ A, $I_{in} = 300$ A, PIV = 100 V

VPI

VPI (tensão de pico inverso) é a máxima tensão que pode ocorrer através do diodo antes de sua destruição. Diodos, diferentemente dos capacitores, são inflexíveis; transientes os destruirão. Não é anormal termos transientes de 400V na linha de 115 VCA, fazendo assim nossos 12,5V alcançar momentaneamente 43V! A ponte retificadora deverá, então, ter VPI mínimo de 50V. Você pode obter uma ponte para 100 VPI com um pouco mais de dinheiro. Lembre-se, segurança custa menos que computadores.

CORRENTE CONTÍNUA

A última consideração é o valor da corrente contínua. Apesar do regulador ter sido projetado para uma saída de 5A, o regulador escolhido fornecerá 7A se curtocircuitado. Este não é o procedimento normal de operação, mas isto pode acontecer. O componente sugerido pode ser a ponte de 12A e 50 VIP. O componente preferido seria o de 12A, 100 VPI ou, por um custo adicional de 15%, um de 27A e 100 VIP. Este último seria até mais do que o suficiente, porém salvará a ponte de diodos caso o capacitor acidentalmente entre em curto. Um transformador de 6A poderá fornecer até 12A em um curco circuito, mas não é como aquele que pode fornecer até 27A. Ambas as escolhas satisfarão o projeto, mas somente uma o salvará contra queima.

O TRANSFORMADOR

Agora vamos considerar o transformador. Nós já determinamos a queda de tensão através de vários componentes. Os valores são usados para calcular a tensão RMS requerida pelo secundário na seguinte fórmula:

$$V_{SEC(RMS)} = \frac{V_C + V_{RIPPLE} + V_{RET}}{\sqrt{2}}$$

V_{RET} = Queda de tensão em cada diodo — (aproximadamente 1V por diodo)

$$= \frac{10 + 2{,}5 + 2{,}0}{1{,}414}$$

$$= 10{,}25 \text{ V}$$

Na prática, um transformador de 10V e 6A estará bem perto.

Os componentes das fontes de + e −12V são escolhidos de maneira similar com a exceção de que a corrente necessária é somente 1A, e uma ponte de 200 VPI é recomendada por causa da configuração particular do retificador. O esquemático final do transformador e da seção de filtro do nosso computador está ilustrado na figura 1.6.

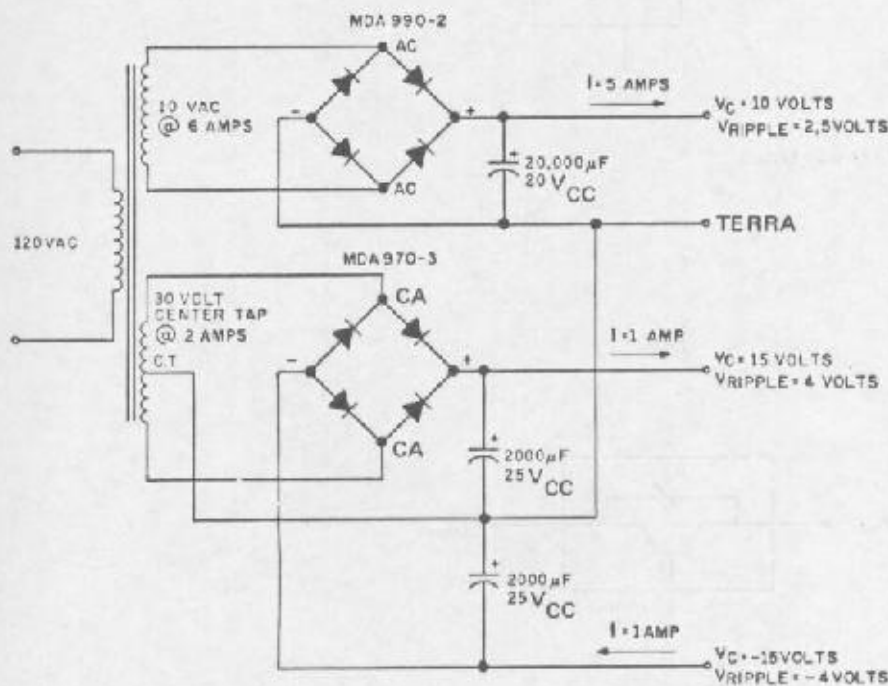



Figura 1.6 Um diagrama esquemático de um transformador e filtro de entrada.

REGULADORES DE TENSÃO

A seção do regulador de tensão de nossa fonte de alimentação é a próxima consideração. Todos os reguladores de tensão possuem a mesma característica: eles convertem uma dada tensão de entrada CC em uma específica tensão de saída estabilizada CC e a mantém apesar de grandes variações da tensão de entrada e da carga de saída. O regulador de tensão típico, como mostrado na figura 1.7, consiste no seguinte:

- Um elemento de referência que fornece uma tensão de referência estável conhecida.
- Um elemento de transformação de tensão que amostra o nível da tensão de saída.
- Um elemento comparador que compara a referência e o nível de saída para gerar um sinal de erro.
- Um elemento de controle que pode utilizar este sinal de erro para gerar uma transformação de tensão de entrada a fim de produzir a saída desejada.

O elemento de controle depende do projeto do regulador e varia muito. O controle determina a classificação dos reguladores de tensão: séries, *shunt*, ou chaveado. Para o regulador série, o elemento de controle regula a tensão de saída pela modulação de um elemento série, normalmente um transistor, fazendo com que este funcione como um resistor variável (figura 1.8). Conforme a tensão de entrada aumente, a resistência em série R_s também aumenta, causando, assim, uma grande queda de tensão sobre esta. Dessa forma, a tensão de saída ($V_{saída}$) é mantida em um nível constante.

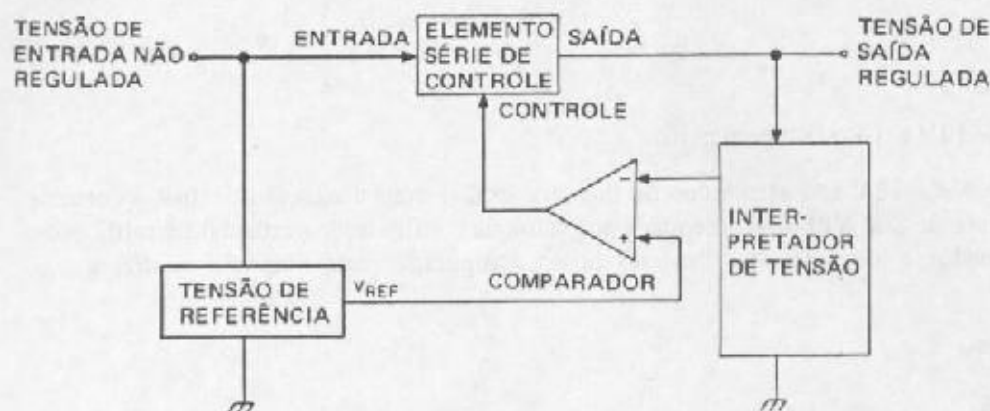



Figura 1.7 O diagrama bloco de um regulador de tensão típico

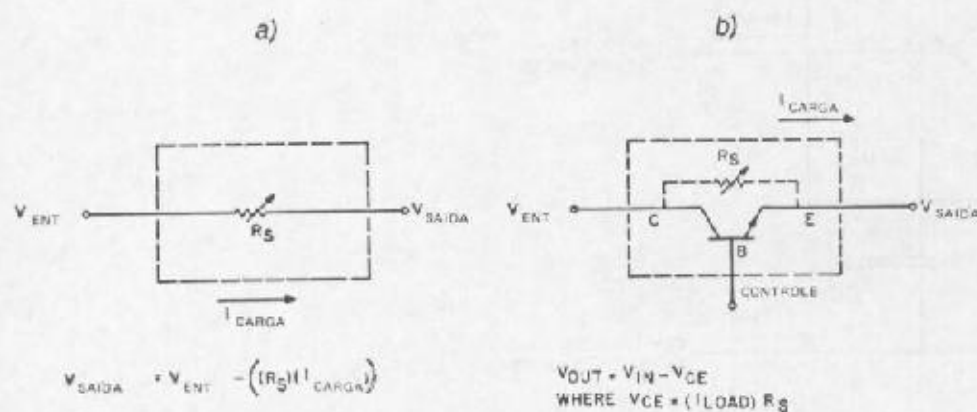



Figura 1.8 Elemento de controle série em um regulador de tensão.
a) O elemento de controle série age como uma resistência variável, R_s.
b) O elemento série é geralmente um transistor.

Para efetuar esse controle do *loop* fechado, está incorporado ao *hardware* um sistema de realimentação e uma comparação de referência. Uma tensão de referência fixa e estabilizada é facilmente produzida por um diodo zener. A corrente produzida é baixa; entretanto, o dispositivo não pode servir como um regulador de potência para si próprio.

O conversor de tensão conectado à saída do elemento de controle série produz um sinal de realimentação que é proporcional à tensão de saída. Em sua forma mais simples, o conversor de tensão é um divisor resistivo. Os dois sinais, de referência e de realimentação, geram a informação necessária para o comparador de tensão a fim de que ocorra a realimentação no *loop* fechado (figura 1.9). A saída do comparador alimenta a base do transistor série, dessa forma a queda de tensão sobre o transistor será mantida em um valor estabilizado quando subtraída da tensão de entrada.

Projetistas modernos, de fonte de alimentação, podem ainda usar componentes individuais para construir um regulador de tensão série, mas muitos reservam este laborioso esforço para aplicações especializadas. O computador descrito aqui necessita apenas de +5V, +12V e −12V. A combinação de temperatura, estabilidade e tolerâncias não podem exceder ±5% de qualquer dos três valores. O modo mais fácil de minimizar riscos é reduzir o número de componentes. Outros projetistas tiveram a mesma idéia e assim foi inventado o regulador de três terminais. A figura 1.10 é o diagrama de bloco de tal dispositivo.

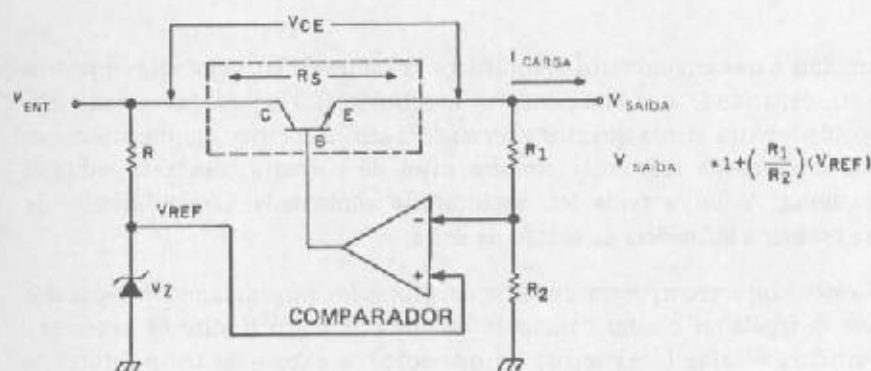



Figura 1.9 Diagrama esquemático de um regulador de tensão série.

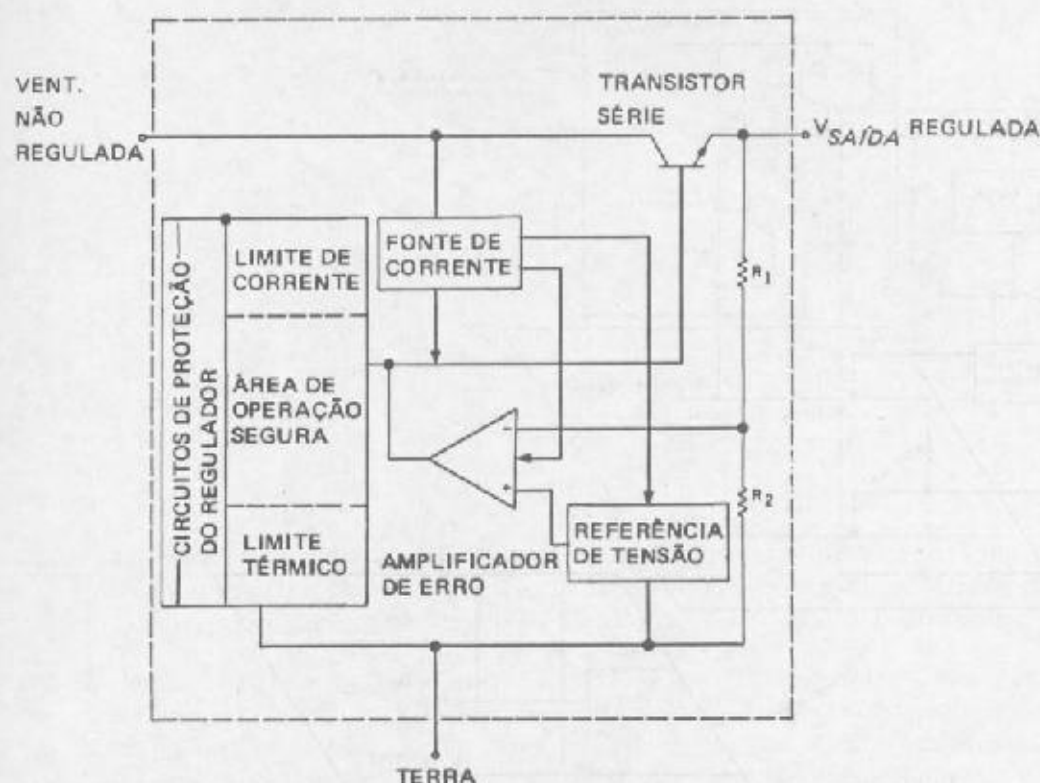



Figura 1.10 Diagrama bloco de um regulador de tensão de três terminais.

Basicamente, um regulador de três terminais incorpora todos os transistores, resistores e diodos em um simples circuito integrado. Embora simples de usar, esses componentes possuem uma estrutura interna muito mais complicada do que o regulador série da figura 1.9. Somente três terminais são necessários em aplicações onde a saída é um valor como: ±5V, ±6V, ±8V, ±12V, ±15V ou ±24V. As três conexões são: CC não regulada do nosso filtro de entrada, uma referência de terra e, finalmente, saída CC regulada.

Em um regulador de três terminais, a tensão de referência é a parte mais importante porque qualquer anormalidade ou perturbação será refletida na saída. Por isso, a referência deve ser estável e sem ruído de perturbação. Componentes mais avançados usam circuitos de referência melhores do que diodos zener. Devido a sua complexidade, tais circuitos só são possíveis em circuitos integrados.

Outra vantagem do regulador de três terminais é que em circuitos monolíticos as fontes de corrente estável podem ser facilmente realizadas devido ao avanço na capacidade dos componentes monolíticos. Também, como no caso anterior, o projetista pode adicionar tantos componentes ativos quantos necessários, sem aumentar significantemente a área do circuito integrado. A operação do circuito de referência em um nível de corrente constante reduz as flutuações oriundas da variação da tensão da linha. Assim, a saída tem estabilidade aumentada. O amplificador de erro também opera em corrente constante para reduzir a influência da tensão da linha.

A consideração mais importante é que esses chips incorporam circuito de proteção, resguardando o regulador de certos tipos de sobrecarga. Eles protegem o regulador contra condições de curto circuito (limite de corrente); condição de alta diferença de tensão entre entrada e saída (área segura de operação); e excessivas temperaturas de junção (limite térmico). Claro que todo este circuito é projetado para proteger o regulador, não o computador.

ESCOLHENDO UM REGULADOR

O regulador de tensão híbrido 5A μA78H05 possui todas as características do regulador monolítico de três terminais (circuito completo de proteção). Cada encapsulamento hermeticamente fechado TO-3 contém um regulador monolítico μA78M05 capaz de alimentar um transistor série discreto Q1 e dois transistores de deteção de curto circuito Q2 e Q3 (veja figura 1.11). O transistor série está montado no mesmo substrato de óxido de berílio do chip regulador, assegurando, assim, transferência térmica ideal entre Q1 e o circuito sensor de temperatura do 78M05.

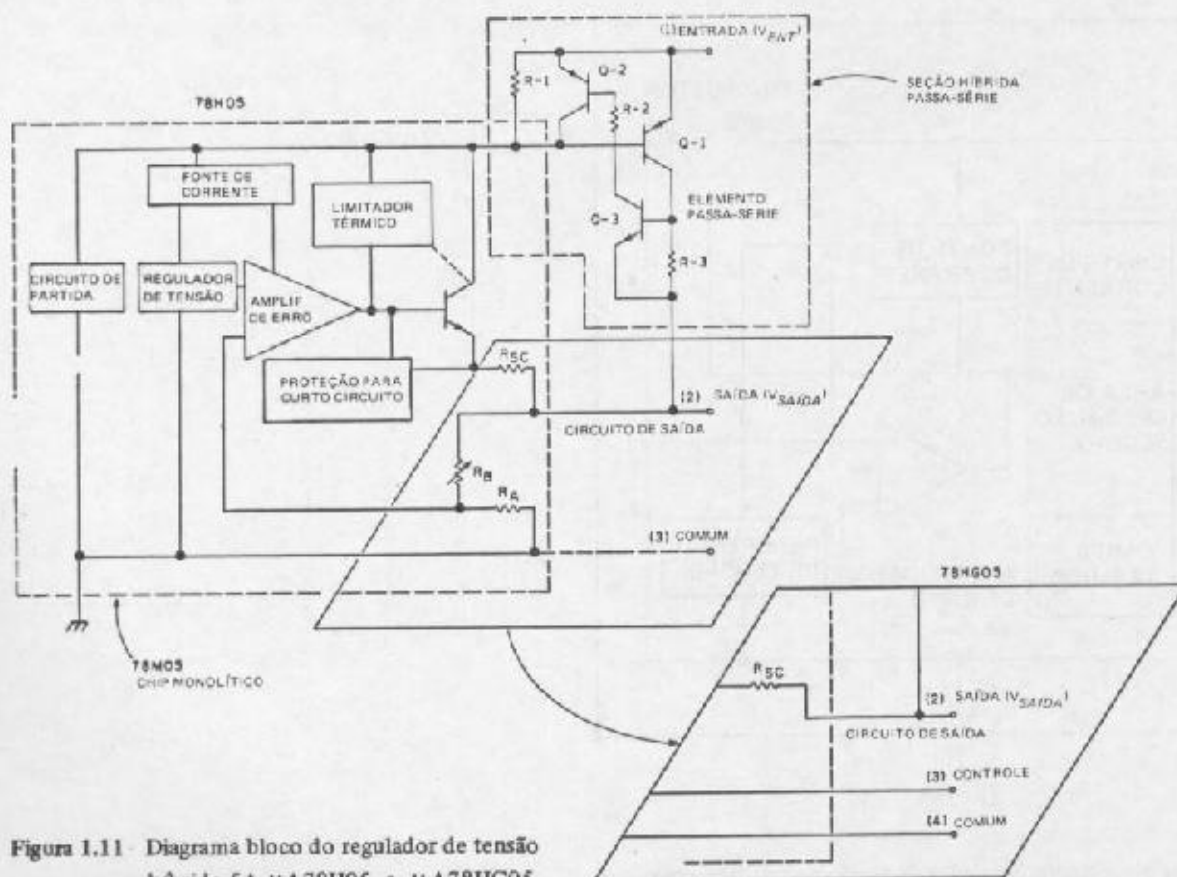



Figura 1.11 Diagrama bloco do regulador de tensão híbrido 5A μA78H05 e μA78HG05.

ELECTRICAL CHARACTERISTICS: T_J = 25°C, I_{OUT} = 2.0 A unless otherwise specified.

CHARACTERISTICS	CONDITIONS	μA78H05C			UNITS
		MIN	TYP	MAX	
Output Voltage	I_{OUT} = 2.0 A, V_{IN} = 10 V	4.8	5.0	5.2	V
Line Regulation	V_{IN} = 8.5 to 25 V		10	50	mV
Load Regulation	10 mA ≤ I_{OUT} ≤ 5.0 A, V_{IN} = 10 V		10	50	mV
Quiescent Current	I_{OUT} = 0, V_{IN} = V_{OUT} + 5.0 V			10	mA
Ripple Rejection	I_{OUT} = 1.0 A, f = 210 Hz, 5.0 V P-P	60			dB
Output Noise	10 Hz ≤ f ≤ 100 kHz, V_{IN} = V_{OUT} + 5.0 V		40		μV_{RMS}
Dropout Voltage	I_O = 5.0 A		3.0		V
	I_O = 3.0 A		2.6		V
Short Circuit Current Limit	V_{IN} = 10 V		7.0		A_{pk}

Figura 1.12 Características elétricas do regulador de voltagem 78H05

O circuito de saída é projetado para que o pior caso de corrente requerida pela base de Q1, somado à corrente através de R2, permaneça sempre abaixo do limite da corrente do 78M05. Resistor R1, em conjunto com Q2 e Q3, servem de sensor e limitador de corrente para proteger o elemento série de um dreno excessivo de corrente.

O projeto foi feito tendo em mente o montador inexperiente. O transistor série é capaz de suportar a corrente de curto circuito com a mesma tensão de entrada permitida pelo 78H05. (Veja figura 1.12 para as características elétricas do 78H05.)

A saída nominal do componente é de 5,0V mas pode variar entre 4,8 e 5,2V. Mesmo assim isto permanece dentro dos 15% de tolerância necessários para fazer o computador funcionar, haverá um problema de queda de tensão na fiação que vai da fonte ao computador. Até 0,5V podem ser perdidos nos conectores e fiação. Lembre-se que com 5A a resistência de 0,1 OHMS pode causar uma queda de 0,5V. Infelizmente o 78H05 é um componente de saída fixa quando referenciado à terra. Se acontecer da saída fornecer 4,8V isto será tudo o que você conseguirá. Mas, como um caso clássico de solução de engenharia, nós podemos enganar o regulador fazendo a referência de terra ajustável. A figura 1.13 mostra o circuito que torna isto possível. Um potenciômetro ligado à fonte de 12V cria uma referência de terra relativa para o 78H05. Se o componente em questão tivesse uma saída de 4,95V, e se nós ajustássemos R1 para um potencial de 0,20V no pino comum do regulador, a saída referenciada à terra mudaria para 4,95 + 0,20, ou 5,15V. Para os aficionados, este circuito em particular também permite que um componente com tensão alta de saída seja reduzida para 5,00V selecionando-se uma voltagem negativa apropriada para o pino de referência.

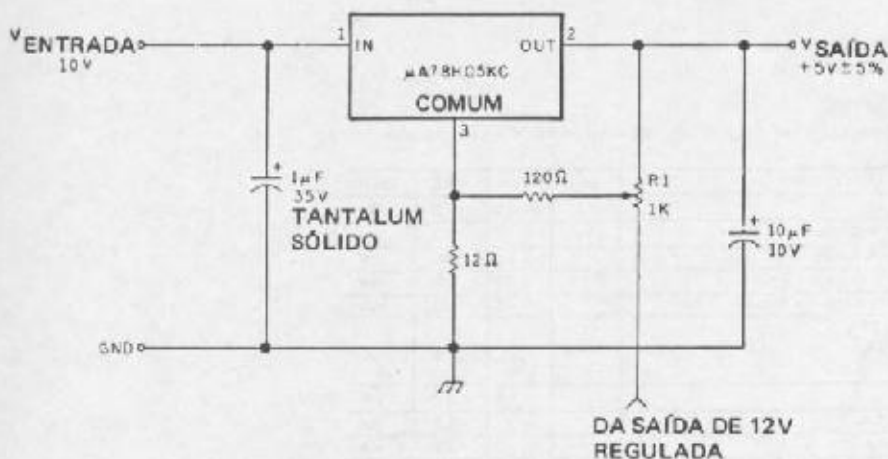



Figura 1.13 Como obter uma saída variável de um regulador de tensão fixo.

Com a fonte de 5V completa, nosso próximo assunto de interesse são as fontes de +12 e −12V. Outros componentes da família 7800 de reguladores satisfarão os requisitos. Os componentes 7812 e 7912 são reguladores positivo e negativo de 1A respectivamente; eles têm as mesmas características de proteção do 78H05. As figuras 1.14 e 1.15 mostram as especificações exatas. Como nós agora estamos lidando com correntes bem menores do que a da

fonte de +5V, existe uma perda muito menor com relação a cabeação e conectores e não é necessário incluir circuitos de ajuste. A figura 1.16 mostra o circuito final da fonte de alimentação. Circuitos reguladores adicionais (figuras 1.17a, b, c e d) são incluídos para demonstrar como a série de reguladores 7800 pode ser usada em nossa aplicação.

Já terminamos? Claro que não. Um exame mais apurado da figura 1.16 mostra dois itens não discutidos anteriormente: dissipadores e proteção de sobretensão. Esses dois assuntos e uma pequena discussão da importância de um layout correto completam o capítulo.

μA7812

ELECTRICAL CHARACTERISTICS: V_{IN} = 19 V, I_{OUT} = 500 mA, −55°C ≤ T_J ≤ 150°C, C_{IN} = 0.33 μF, C_{OUT} = 0.1 μF, unless otherwise specified.

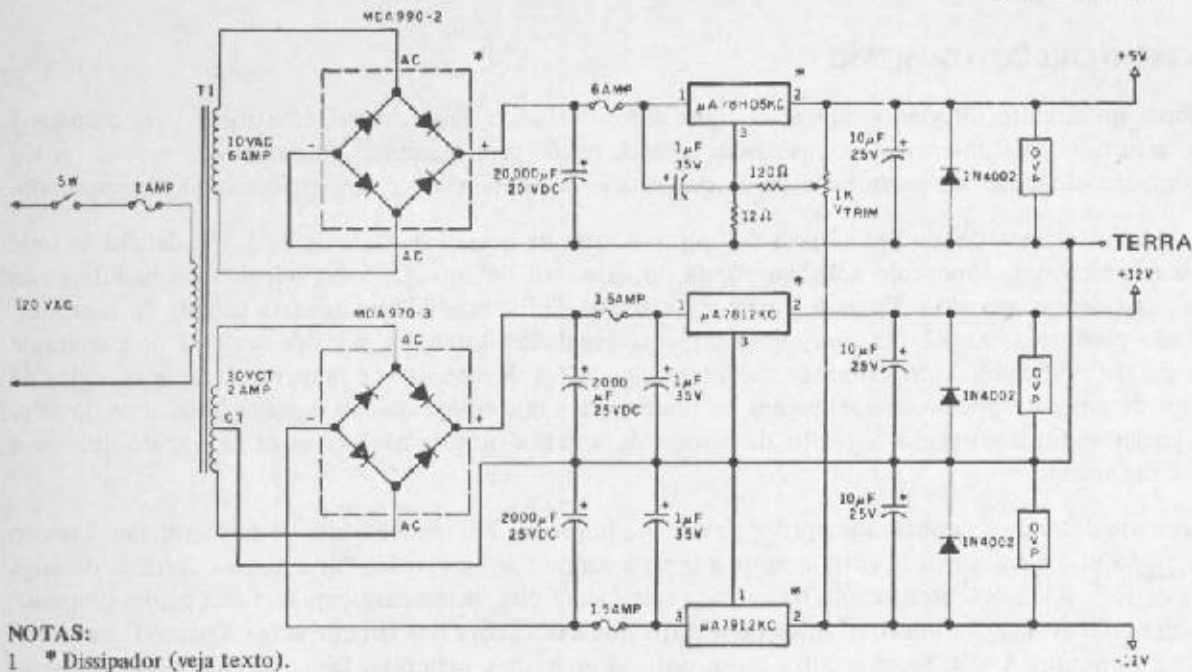
CHARACTERISTICS		CONDITIONS		MIN	TYP	MAX	UNITS
Output Voltage		T_J = 25°C		11.5	12.0	12.5	V
Line Regulation		T_J = 25°C	14.5 V ≤ V_{IN} ≤ 30 V		10	120	mV
			16 V ≤ V_{IN} ≤ 22 V		3.0	60	mV
Load Regulation		T_J = 25°C	5 mA ≤ I_{OUT} ≤ 1.5 A		12	120	mV
			250 mA ≤ I_{OUT} ≤ 750 mA		4.0	60	mV
Output Voltage		15.5 V ≤ V_{IN} ≤ 27 V 5 mA ≤ I_{OUT} ≤ 1.0 A P ≤ 15 W		11.4		12.6	V
Quiescent Current		T_J = 25°C			4.3	6.0	mA
Quiescent Current Change	with line	15 V ≤ V_{IN} ≤ 30 V				0.8	mA
	with load	5 mA ≤ I_{OUT} ≤ 1.0 A				0.5	mA
Output Noise Voltage		T_A = 25°C, 10 Hz ≤ f ≤ 100 kHz			8	40	μV/V_{OUT}
Ripple Rejection		f = 120 Hz, 15 V ≤ V_{IN} ≤ 25 V		61	71		dB
Dropout Voltage		I_{OUT} = 1.0 A, T_J = 25°C			2.0	2.5	V
Output Resistance		f = 1 kHz			18		mΩ
Short Circuit Current		T_J = 25°C, V_{IN} = 35 V			0.75	1.2	A
Peak Output Current		T_J = 25°C		1.3	2.2	3.3	A
Average Temperature Coefficient of Output Voltage		I_{OUT} = 5 mA	−55°C ≤ T_J ≤ +25°C			0.4	mV/°C/
			+25°C ≤ T_J ≤ +150°C			0.3	V_{OUT}

Figura 1.14 Características elétricas do regulador de tensão μA7812.

μA7912

ELECTRICAL CHARACTERISTICS: V_{IN} = −19 V, I_{OUT} = 500 mA, C_{IN} = 2 μF, C_{OUT} = 1 μF, −55°C ≤ T_J ≤ 150°C, unless otherwise specified.

CHARACTERISTICS		CONDITIONS		MIN	TYP	MAX	UNITS
Output Voltage		T_J = 25°C		−11.5	−12.0	−12.5	V
Line Regulation		T_J = 25°C	−14.5 V ≤ V_{IN} ≤ −30 V		10	120	mV
			−16 V ≤ V_{IN} ≤ −22 V		3.0	60	mV
Load Regulation		T_J = 25°C	5 mA ≤ I_{OUT} ≤ 1.5 A		12	120	mV
			250 mA ≤ I_{OUT} ≤ 750 mA		4.0	60	mV
Output Voltage		−15.5 V ≤ V_{IN} ≤ −27 V 5 mA ≤ I_{OUT} ≤ 1.0 A p ≤ 15 W		−11.4		−12.6	V
Quiescent Current		T_J = 25°C			1.5	3.0	mA
Quiescent Current Change	with line	−15 V ≤ V_{IN} ≤ −30 V				1.0	mA
	with load	5 mA ≤ I_{OUT} ≤ 1.0 A				0.5	mA
Output Noise Voltage		T_A = 25°C, 10 Hz ≤ f ≤ 100 kHz			25	80	μV/V_{OUT}
Ripple Rejection		f = 120 Hz, −15 V ≤ V_{IN} ≤ −25 V		54	60		dB
Dropout Voltage		I_{OUT} = 1.0 A, T_J = 25°C			1.1	2.3	V
Peak Output Current		T_J = 25°C		1.3	2.1	3.3	A
Average Temperature Coefficient of Output Voltage		I_{OUT} = 5 mA, −55°C ≤ T_J ≤ 150°C				0.3	mV/°C/V_{OUT}
Short Circuit Current		V_{IN} = −35 V, T_J = 25°C				1.2	A

Figura 1.15 Características elétricas do regulador de tensão μA7912.

NOTAS:

1. * Dissipador (veja texto).
2. 1 μF/35V capacitor de *tantalum*
3. Note que existe uma designação diferente de pinagem entre o 7812 e o 7912.
4. O fusível é colocado à entrada do regulador e entre o capacitor de filtro e a ponte de diodos.

Figura 1.16 Diagrama esquemático da fonte completa para o computador.

a)
ENTRADA DESREGULADA
2N4398
6Ω 3W
μA78M05
+5V
0.33μF 100V
10μF 10V
$I_{SAÍDA} \geq 5$ AMPÈRES

b)
ENTRADA DESREGULADA
R_{SC}
2N4398
2N6124
6Ω 3W
μA78M05
+5V
0.33μF 100V
10μF 10V
$I_{SAÍDA} = \frac{V_{BE}}{R_{SC}}$

c)
REGULADOR DE TENSÃO VARIÁVEL
ENTRADA DESREGULADA 10 – 33V
μA7805
0.33μF 100V
LM741
10K
7 A 30V AJUSTÁVEIS @1 AMP
1μF 35V

d)
ENTRADA DESREGULADA
μA7812
0.33μF 100V
LM741
2N6124
4.7K
+12V SAÍDA DE 1 AMPÈRE
COMUM
-12V SAÍDA DE 1 AMPÈRE
ENTRADA DESREGULADA
REGULADOR DUPLO SEGUIDOR DE ± 12V

Figura 1.17 Diagramas esquemáticos adicionais que mostram como os reguladores da família 7800 podem ser usados.
a) Regulador de alta corrente usando o regulador 78M05 de 500 MA de três terminais.
b) Regulador de tensão de alta corrente com proteção de curto circuito, uma versão melhorada da figura 1.17a.
c) Usando um regulador de tensão de + 5V 7805 para obter tensões de saída mais altas.
d) Regulador de tensão dupla seguidora de + 12V.

A IMPORTÂNCIA DO CIRCUITO IMPRESSO

Os reguladores em circuito integrado empregam transistores de banda larga em sua construção para otimizar a resposta. Como resultado eles devem ser compensados propriamente para assegurar uma operação estável em elo fechado. Sua compensação pode ser perturbada por capacitâncias e indutâncias de um circuito impresso impróprio.

A figura 1.18a ilustra uma distribuição típica dos componentes da nossa fonte e a figura 1.18b detalha as áreas que podem causar problemas. Uma colocação imprópria do capacitor de entrada pode induzir uma ondulação na tensão de saída. Isto ocorre quando o fluxo de corrente de entrada influência a linha de terra comum do regulador. A queda de tensão produzida em R2 faz com que a saída do regulador flutue. Os picos de corrente do circuito de entrada (composto do retificador e do capacitor de filtro) podem ser de dezenas de ampères durante os ciclos de carga. Esses picos de corrente podem causar quedas de tensão em pinos compridos ou ligações feitas com fio fino. Eles podem degradar o funcionamento a ponto da tensão de entrada própria não ser mantida, exceto durante a operação com baixa corrente.

O elo da corrente de saída é também susceptível ao circuito impresso. Em um regulador de três terminais, a tensão de saída fixa $V_{saída}$(REG) é referenciada entre a saída e o pino comum do integrado. Por causa da corrente de carga que flui através de R2', R3' e R4' assim como na carga propriamente dita, ocasionam perdas. Essas perdas de tensão combinadas podem reduzir $V_{saída}$ a um nível intolerável. Note que a terra para este circuito está no ponto C, enquanto a carga está entre os pontos A e B. Se uma outra carga, por exemplo, mais memória, for conectada à fonte entre os pontos A e C teria um $V_{saída}$ diferente. Ajustar o potenciômetro pode ser perigoso, pois é possível ter uma carga dentro da tolerância e a outra com uma tensão acima ou abaixo. Um último ponto a considerar é que R4' prejudica o funcionamento do regulador porque reduz continuamente a tensão de saída quando a corrente de carga aumenta.

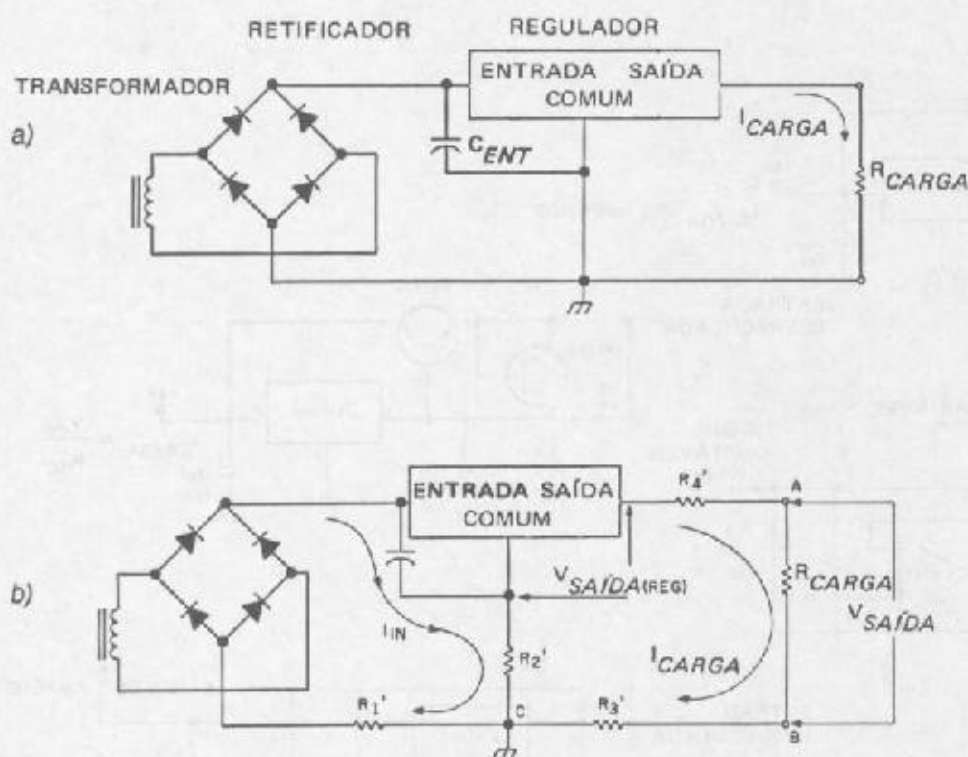



Figura 1.18 Desenho esquemático típico de uma fonte e seus problemas associados.
a) Desenho esquemático típico.
b) Problemas causados pelo desenho da figura 1.18a.

A figura 1.19 mostra o diagrama em bloco de um desenho esquemático certo. Todos os caminhos de corrente devem ser feitos com fio grosso para minimizar a resistência e as quedas de tensão resultantes. Você notará agora que os caminhos de corrente da entrada e da saída estão separados realmente. Note que os fios vindos do retificador vão diretamente para o capacitor e que os dois fios do capacitor é que alimentam o resto do circuito.

Se você seguir esta convenção, os erros induzidos do circuito de entrada poderão ser eliminados. Finalmente, nós precisamos discutir o conceito de um único ponto de terra. Um ponto na fonte deve ser designado como terra; as terras de todos os outros circuitos deverão estar ligadas a este ponto.

Na prática a melhor maneira para se implementar esta conexão de terras é usar um terminal de metal ou vários fios grossos soldados juntos. O terminal é uma barra de terra com uma resistência baixa de tal maneira que em qualquer tensão medida entre o ponto A e qualquer ponto desta barra não será sentida nenhuma variação. Uma outra barra de + 5V deve ser ligada à saída da fonte, para que a distribuição da tensão através dos circuitos seja consistente.

Use fios grossos para a alimentação. Mesmo que um fio com resistência zero não seja tão fácil de ser encontrado, lembre-se que não há nada melhor que um fio bem grosso.

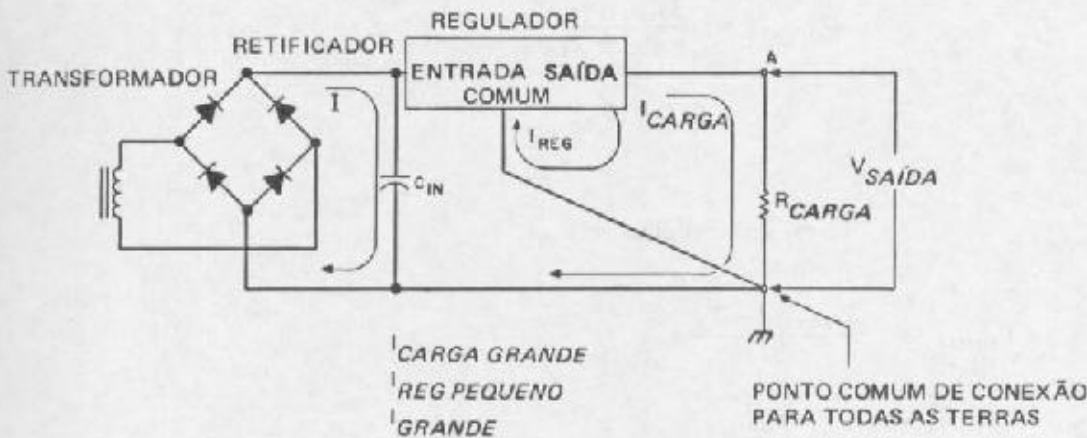



Figura 1.19 Diagrama em bloco da disposição correta dos componentes na fonte.

CONSIDERAÇÕES TÉRMICAS

Você acabou de construir a fonte, ligou a força e tudo está funcionando. Depois de alguns minutos, alguma coisa acontece e o computador repentinamente pára de funcionar. Naturalmente, você começa a olhar e tocar as coisas. Eventualmente você irá tocar no circuito regulador e queimar o seu dedo. Quando os reguladores não são devidamente esfriados existe um circuito protetor interno que desliga sua saída.

O problema se apresenta maior quando se está usando circuitos integrados que usam duas ou mais tensões de alimentação. A queda de uma delas pode causar danos permanentes a eles. Isto nunca acontecerá se a dissipação da fonte for limitada e métodos corretos de dissipação forem empregados.

O primeiro passo é testar a dissipação do nosso projeto com a especificada pelo componente. Na prática, a potência é expressa em WATTS, que é VOLT vezes Ampère:

$$P_D = E \times I$$

No nosso regulador de 5V nós temos $V_c = 10V$ e $V_{Pico} = 12{,}5V$ em 5A.

$$\begin{aligned} P_{D(NOM)} &= (V_C - V_{SAÍDA}) \times 5\,A \\ &= (10 - 5) \times 5 \\ &= 25\,W \\ P_{D(PICO)} &= (V_{PICO} - V_{SAÍDA}) \times 5\,A \\ &= (12{,}5 - 5) \times 5 \\ &= 37{,}5\,W \\ P_{D(MÉDIO)} &= \frac{37{,}5 + 25}{2} = 31{,}25\,W \end{aligned}$$

Conclui-se que, sob as condições de carga máxima, mais ou menos 30W de calor serão produzidos pelo 78H05. A dissipação especificada pelo componente é de 50W em 25°C, e de 30W em 75°C.

Embora a dissipação interna seja limitada, a temperatura da junção deve ser mantida abaixo da máxima temperatura especificada para que o componente funcione.

Para se calcular o dissipador necessário, existem equações características a resolver.

A seguir mostraremos os dados térmicos necessários aos cálculos:

$$\theta_{JC} \text{ Típico} = 2{,}0 \quad \theta_{JC} \text{ Máximo} = 2{,}5$$
$$\theta_{JA} \text{ Típico} = 32 \quad \theta_{JA} \text{ Máximo} = 38$$

$$P_{D(MAX)} = \frac{T_{J(MAX)} - T_A}{\theta_{JC} + \theta_{CA}}$$

$$\theta_{CA} = \theta_{CS} + \theta_{SA}$$

Resolvendo para T_{j1}

$$T_J = T_A + P_D(\theta_{JC} + \theta_{CA})$$

Sem um dissipador

$$P_{D(MAX)} = \frac{T_{J(MAX)} - T_A}{\theta_{JA}}$$

$$T_J = T_A + P_D\theta_{JA}$$

onde, T_J = temperatura da junção
T_A = temperatura ambiente
P_D = dissipação de potência
θ_{JC} = resistência térmica entre a junção e o invólucro
θ_{JA} = resistência térmica abre a junção e o ambiente
θ_{CA} = resistência térmica do ambiente para o invólucro
θ_{CS} = resistência térmica do invólucro para o dissipador
θ_{SA} = resistência térmica do dissipador para o ambiente

$$\theta_{JA} = \frac{T_J - T_A}{P_D} = \frac{125^\circ C - 25^\circ C}{31{,}25\ W} = 3{,}2^\circ C/W$$

Como θ_{JA} é menor do que o θ_{JA} da especificação do componente, um dissipador é realmente necessário, e um dissipador do tipo – TO-3 de 3,2°C/W é o mínimo necessário.

Antes de você especificar um dissipador para o 78H05, note que existem mais dois reguladores e mais duas pontes retificadores que necessitarão de dissipadores. Cada um dos reguladores de 12V terá aproximadamente 5W de dissipação. A ponte de diodos associada à ponte de 5V (lembre-se da queda de 2V) dissipa 10W enquanto a outra 2W. Por isso qualquer dissipador na ponte terá que dissipar mais de 50W.

QUAL O MÉTODO PRÁTICO PARA SE ESCOLHER DISSIPADORES ?

Escolher um dissipador pode ser fácil ou difícil dependendo da sua experiência. Nós já sabemos que precisamos de um dissipador de 50W. É fácil comprar um especificado para 50W em uma loja, o que resolverá o problema. Isto significa que se aplicarmos estes 50W através de um transistor a este dissipador com uma temperatura ambiente de 25°C, a temperatura do dissipador irá aos 100°C.

Nós não podemos esquecer que os fabricantes, nas especificações, sempre se referem ao limite máximo da temperatura de junção, mas não em termos de manter o invólucro frio o suficiente para ser tocado. Pessoalmente eu detesto fontes quentes e vermelhas. Obter um dissipador que dissipe 50W e mantenha a temperatura em 60-70°C significa conseguirmos um que dissipe 200 a 300W. Lembre-se, dissipadores são grandes e caros.

A solução mais simples é a melhor. Eu prefiro resfriamento por ar. Coloque os 50W em um dissipador de 100W e o seu dinheiro em um bom ventilador. Você pode continuar os seus cálculos e determinar quantos centímetros quadrados serão necessários, mas o efeito de soprar um pouco de ar sobre o dissipador multiplica a sua capacidade enormemente.

PROTEÇÃO DE SOBRETENSÃO

O último ponto a ser visto na fonte é a proteção de sobretensão.

Os reguladores da maneira como são projetados já se protegem diminuindo a tensão de saída. A chance de danos aos componentes do computador por tensão baixa é mínima comparada à sobretensão.

PST = proteção sobretensão

PST + 5V		
D_1	5,6V	1N4734
SCR_1	50V 25A	2N682
Fusível	6 Amp	rápido

PST + 12 V		
D_1	13V	1N4743
SCR_1	50V 8A	2N4441
Fusível	1,5 Amp	rápido

Os semicondutores usados no circuito PST + 12V são usados com polaridade inversa para o PST – 12V.

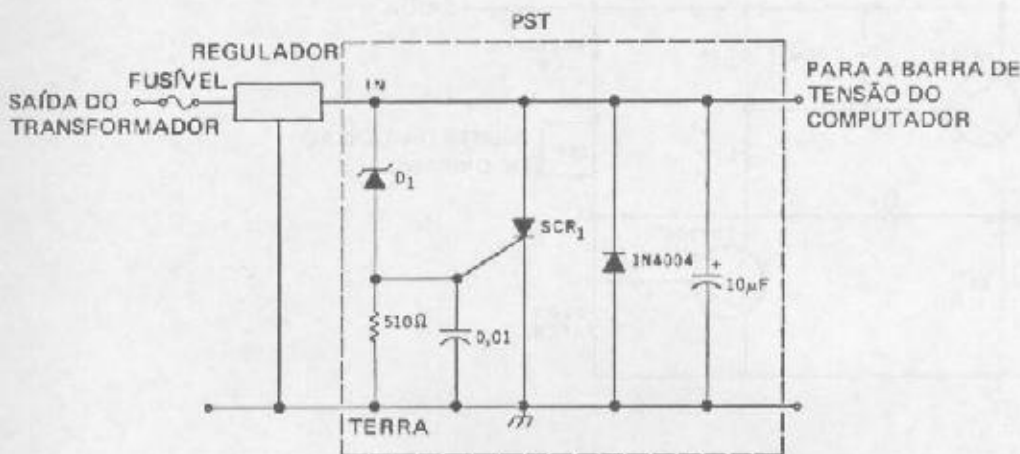



Figura 1.20 Circuito de proteção de sobretensão.

O circuito da figura 1.20 é um circuito simples de proteção de sobretensão (PST) que pode ser usado, como mostrado, para as fontes de + 5V e 12V. Os componentes para as diferentes fontes estão listados nas tabelas da figura 1.20. Você notará que os fusíveis foram especificados com valores maiores do que havíamos projetado. Os fusíveis são para o PST e não para proteger os reguladores. Infelizmente, a característica dos fusíveis rápidos não deixa que passem 5A se é um fusível de 5A, mas sim para abrir em 5A. O fusível deve ter uma especificação maior para que permita passar os 5A.

Como a corrente de curto circuito do 78H05 é de 7A, o SCR_2 quando disparar fará com que o fusível se abra. As figuras 1.21 e 1.22 mostram circuitos um pouco mais complexos de PST que também podem ser usados.

Como funciona um PST? Ele monitora uma barra de tensão em particular e curtocircuita-a se esta estiver acima de uma tensão predeterminada. Circuitos de PST podem ser projetados para disparar com tensão de 1m V acima da que usamos. Estes circuitos não são só complicados, mas podem também criar problemas adicionais através de disparos acidentais. As falhas que mais comumente ocorrem estão relacionadas ao regulador que entrar em curto ou a duas saídas de fonte que se juntarem, por exemplo, a de + 5V, e + 12V. Em ambos os casos, a tensão de saída aumentará. Estas tensões subindo acima do valor do diodo zener fará com que flua corrente na porta do SRC. Esta corrente irá disparar o SCR que fará com que o fusível se abra. Ambos os fusíveis abrirão se juntarmos as fontes de + 5V e + 12V. O circuito de teste da figura 1.23 mostra o que acontece quando juntarmos a fonte de + 5V com a de + 12V.

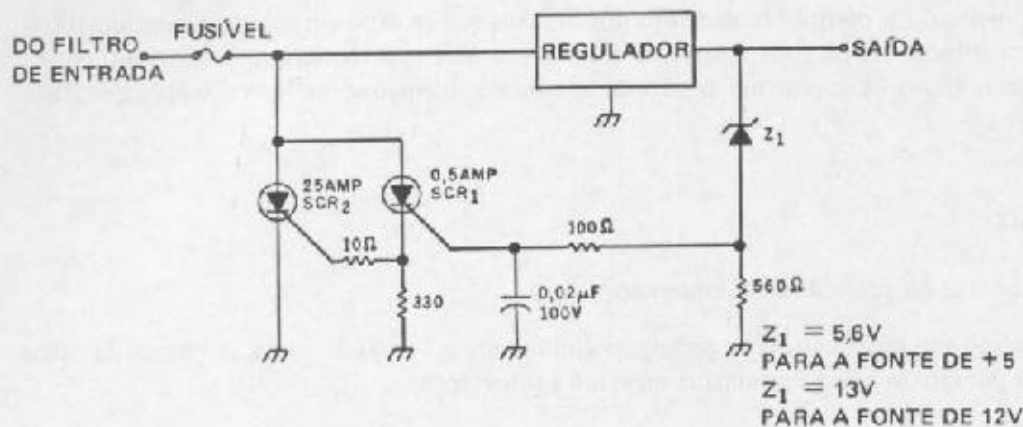



Figura 1.21 Diagrama esquemático de um circuito de proteção de sobretensão mais complexo. O circuito de corte da tensão está localizado perto do fusível enquanto o sensor do PST (Z_1) está localizado na saída do regulador. Esta é uma melhor localização dos componentes se o sensor e o cortador puderem ser separados adequadamente. O sensor de baixa corrente Z_1 dispara o SCR_1 numa condição de sobretensão. O SCR_1 por seu lado dispara o SCR_2.

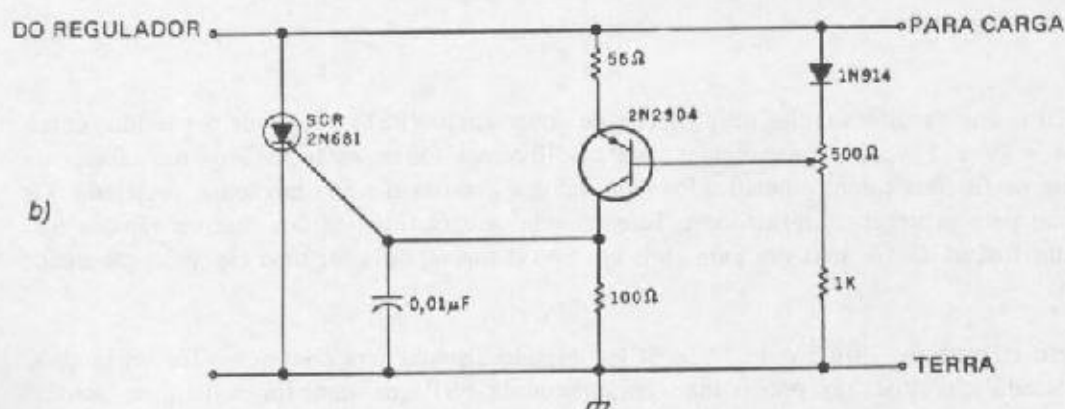
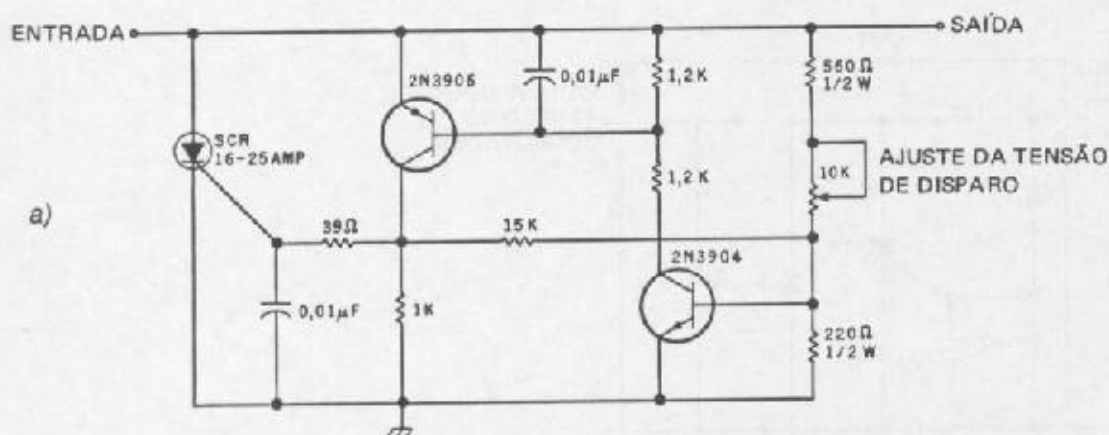



Figura 1.22 Diagrama esquemático de circuitos de proteção de sobretensão ajustáveis.
a) PST ajustável com um amplificador interno de corrente para disparar a porta do SCR.
b) Um circuito alternativo mais simples.

O SCR nunca permite que a linha de + 5V vá para 12V, curtocircuitando-a imediatamente para a terra. Trocando o fusível por um resistor de 220 OHM pode-se fazer o teste repetidamente.

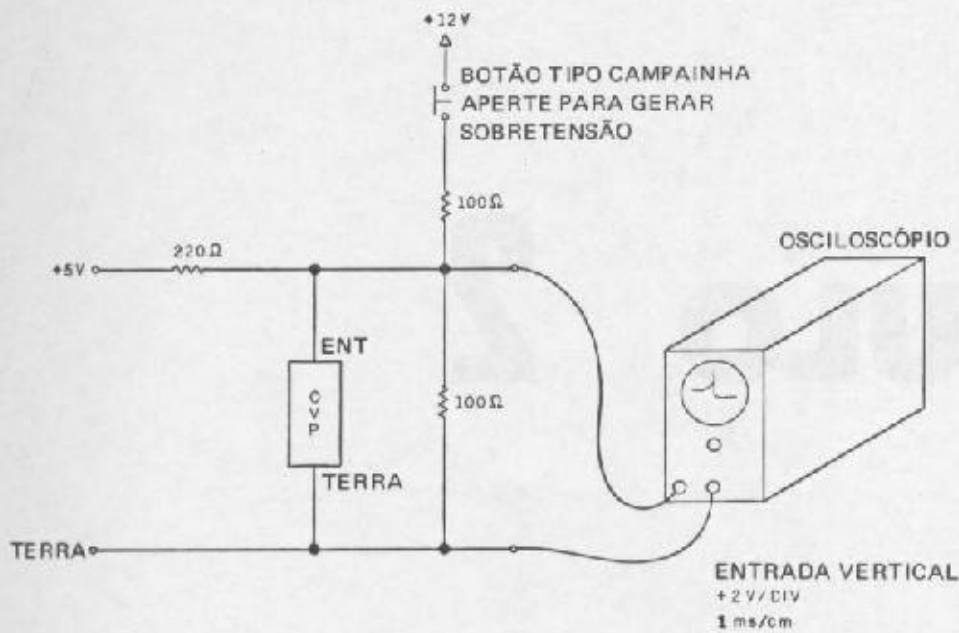



Figura 1.23 Circuito de teste para demonstrar a ação do PST.

CAPÍTULO 2

O BÁSICO DO PROCESSADOR CENTRAL

Existem muitos microprocessadores diferentes no mercado e enquanto a nomenclatura das instruções for um pouco diferente para cada um, os processos lógicos básicos de computação serão similares em todos. Uma regra para se lembrar no futuro, durante uma discussão sobre a capacidade de dois computadores é que "um computador é um computador". Não quero com isto dizer que todos sejam iguais, mas existem coisas parecidas e eu não gostaria de passar o resto de minha vida analisando grupo de instruções e detalhes de ligações antes de escolher um.

Uma vez almocei com um projetista de um dos mais vendidos sistemas de computadores pessoais do mercado. Milhares de computadores foram vendidos. Nossa conversa girou em torno do preço – funcionalidade do sistema.

Eu tinha em mente usar um grupo de engenheiros durante meses no projeto, reduzindo o número de componentes e analisando as instruções para determinar o menor tamanho de memória necessário. Na verdade ao meu amigo projetista foram dados dois meses para fazer alguma coisa que pudesse ser fabricada. As únicas perguntas dos investidores foram o preço e a facilidade de se achar os componentes. Como era um entusiasta de computadores pessoais ele simplesmente fez um computador em torno de um microprocessador que ele já possuía. A facilidade do projeto deveu-se à avançada arquitetura encontrada no processador central, mas nenhuma facilidade em termos de programação em linguagem de máquina era dada ao usuário. Existia apenas um interpretador de alto nível em BASIC considerado do ponto de vista técnico um computador tipo caixa-preta. Ele poderia ter usado qualquer microprocessador. Infelizmente o hobista que está montando um microcomputador, e que não estará fazendo uma caixa-preta, tenta obter um componente que está em algum lugar entre capacidade e desempenho. Ele tem de fazer toda a fiação à mão e certamente estará interessado em um projeto eficiente. É fato que alguns dos microprocessadores requerem circuitos periféricos muito caros. A melhor configuração deve ser a que esteja entre a complexidade do circuito, facilidade de teste, e preço dos componentes.

ARQUITETURA DO MICROPROCESSADOR

A arquitetura interna do microprocessador determina os componentes que serão necessários para o sistema do microcomputador. Talvez como início o melhor seja falar brevemente sobre as diferenças de arquitetura.

Definição: Um microcomputador é uma máquina lógica que manipula números binários (DADOS) e processa esta informação seguindo uma seqüência de passos de programa referenciada como instruções.

Todos os microcomputadores, como todos os computadores têm os seguintes fatores:

1. Entrada – Facilidades devem existir para permitir a entrada de dados ou instruções.
2. Memória – O programa deve ser armazenado antes e após a execução e deve-se poder guardar os resultados da computação.
3. Unidade de aritmética e lógica – Executa as operações aritméticas com dados de entrada ou os que estavam armazenados.
4. Seção de controle – Toma as decisões de acordo com o fluxo de programa e controle do processo baseado em estados internos do resultado das operações aritméticas.
5. Saída – Os resultados são entregues ao usuário ou guardados em um meio apropriado.

O microprocessador é um único circuito integrado em volta do qual um microcomputador é construído. *O microprocessador é um componente; o microcomputador é um sistema.* Nas suas formas menos complexas, os microprocessadores incluem só as funções dos itens três e quatro e dependem de componentes externos ligados à via de dados para executarem outras tarefas. A figura 2.1 mostra o diagrama de blocos básicos de um microcomputador de 8 bits e mostra as ligações das vias e os elementos de suporte. O computador na figura 2.1 usa seis vias separadas: endereço de memória, saída e entrada de dados da memória, endereço de entrada/saída, e saída e entrada de dados. O microprocessador contém um processador central que consiste de um circuito que gera os endereços apropriados para a memória de entrada/saída e interpreta as instruções que são executadas nesta unidade. O processador central também contém a ULA (Unidade de Lógica e Aritmética), a qual executa as operações lógicas e aritméticas dos dados. É composto por uma seção de controle que governa as operações do computador e dos vários registradores de dados usados para manipulação e guarda de dados e instruções.

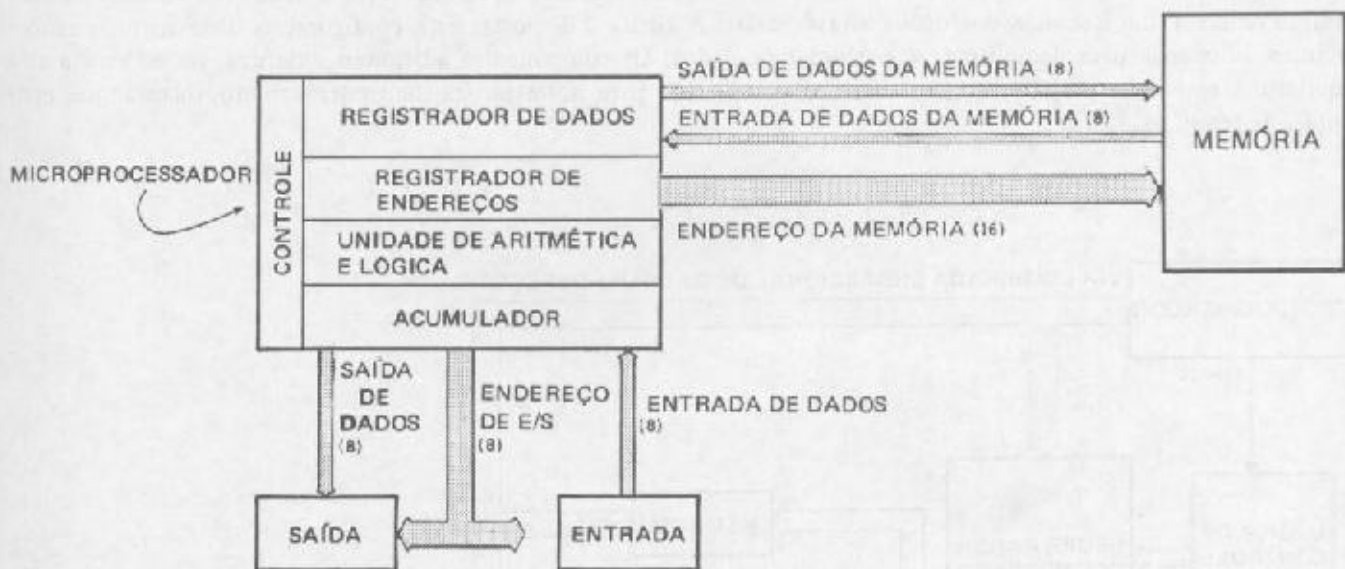



Figura 2.1 Diagrama em bloco de um microcomputador ilustrando o conceito de vias. Os números em parênteses são a quantidade de fios necessários a executar as funções das vias para um microprocessador de 8 bits.

Em realidade poucos microprocessadores comportam seis vias separadas. O número de pinos que seriam necessários no CI está fora de questão. Em vez de reduzir o número de pinos, os fabricantes combinam as vias de entrada e saída de dados em uma só, fazendo-a "bidirecional". Durante uma instrução de saída os dados fluem do microprocessador para o componente de saída e vice-versa durante uma instrução de entrada. Para diminuir ainda mais o número de pinos do processador central, a via de endereços da memória pode também servir como via de endereços dos componentes de entrada/saída. Durante as instruções de entrada/saída o endereço presente nas linhas de endereço referenciam um componente particular de entrada/saída. A configuração reduzida é apresentada na figura 2.2.

O conceito de duas vias é fácil de ser entendido, e do ponto de vista eletrônico, fácil de ser utilizado. As vias são multiplexadas tanto em tempo como em função. Isto é, durante as operações de memória, os bits na via de endereços referem-se a uma posição de memória, e os dados na via de dados representam o conteúdo da memória. A direção do fluxo de dados (do e para o processador central) é controlada dentro do microprocessador. As atividades de entrada/saída são executadas da mesma maneira. Durante estas instruções, os dados de entrada ou saída e o endereço do componente ocupam as vias.

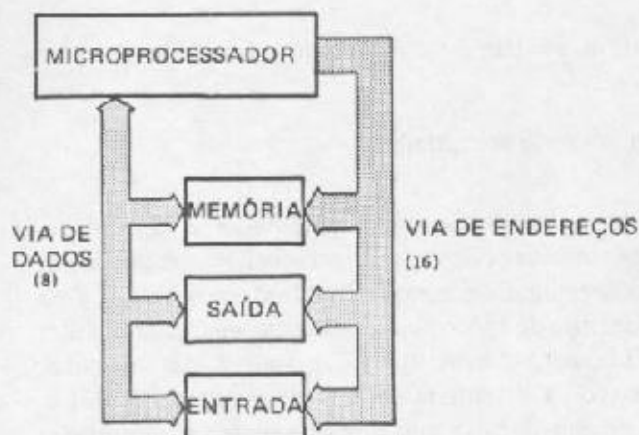



Figura 2.2 Diagrama em blocos de um microcomputador usando a técnica de vias bidirecionais multiplexas para reduzir o número de pinos.

O número de fios das vias podem ser ainda mais reduzidos combinando endereços e dados nas mesmas linhas e multiplexando a transferência dos dados através delas. A figura 2.3 mostra esta configuração. Este método requer circuitos adicionais para demultiplexar e guardar os dados. Os componentes adicionais externos, necessários a esta arquitetura, tornam o projeto um tanto quanto complicado para hobistas. Existem outros microprocessadores mais simples de serem usados.

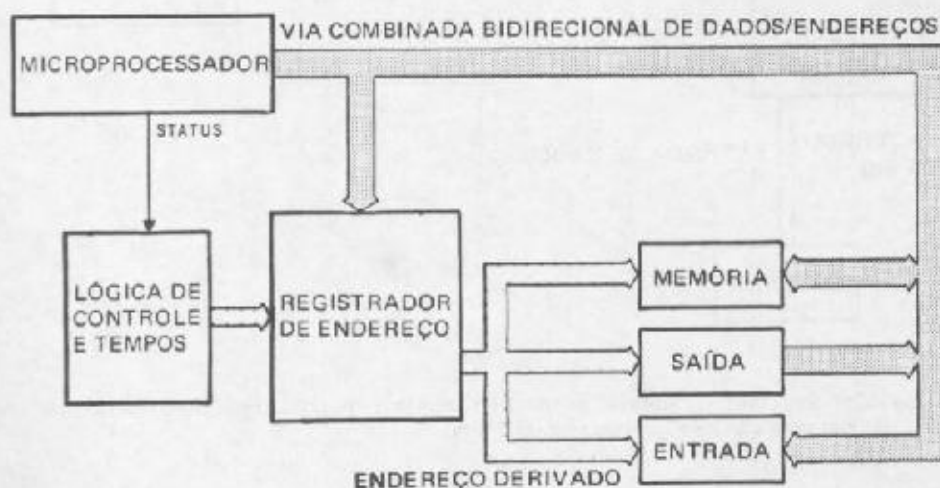



Figura 2.3 Diagrama em bloco de um microcomputador usando uma única via bidirecional multiplexada para as funções de memória e entrada/saída.

Quando estiver montando em vez de comprando um computador pessoal, os seguintes critérios devem ser cuidadosamente considerados:

1. Complexidade do circuito – Mantenha o número de componentes a um mínimo possível. Quanto mais componentes em um projeto, maior a chance de erros na fiação.

2. Custo – Custo é importante, mas não deve ser o requisito de maior importância. Qualquer função do microprocessador pode ser simulada usando integrados de pequena escala de integração, entretanto, custos indiretos resultantes do uso de 200 integrados em vez de 3 ou 4 integrados (de larga escala de integração) teria um resultado ao contrário. Por outro lado, na indústria de semicondutores, densidade significa cruzeiros. Quanto mais funções um componente fornece, e menos componentes são necessários para executar essas tarefas, maior o preço. O computador que será descrito neste livro condiz com esta filosofia. Ele usa uma combinação de integrados de larga escala com integrados de pequena escala para produzir um computador que o hobista poderá realmente construir, testar e usar.

3. Eficácia e compatibilidade de software – A construção do hardware do microcomputador é somente metade do trabalho. Ele deve ser programado para executar trabalho proveitoso. Inicialmente, teremos de codificar e montar os nossos próprios programas. Eventualmente, entretanto, a necessidade de programas muito extensos poderá surgir, neste caso não será fácil montá-lo manualmente. O usuário deverá contar com um programa montador em uma máquina maior. O programa montador poderá, certamente, ser compilado com as instruções disponíveis (set de instruções) do microcomputador.

Uma consideração suplementar é que adeptos do computador pessoal estão sempre trocando de software. É possível converter programas para rodarem em qualquer processador central, mas o esforço seria o mesmo de escrevê-lo inteiramente desde o início. Isto anula o propósito de troca de software. O proprietário de computador pessoal deve escolher um microprocessador que seja de alguma forma compatível com os computadores existentes no mercado. Minha proposição de que todos os computadores são semelhantes é teoricamente verdade, mas um livro de como construir um computador esotérico, único da espécie, é um pouco fora de propósito.

Cada critério pode ser analisado e respondido individualmente, mas devemos dar algum crédito aos fabricantes de computadores pessoais por terem feito algo por nós já pensado. O fato é que muitos computadores pessoais em uso possuem processador central compatível. Para ser compatível com o software existente e para se ter à disposição suficiente documentação, deve-se considerar a escolha de um processador central em uso comercial. Os quatro microprocessadores mais usados são:

1. Intel 8080A
2. Motorola 6800
3. MOS Technology 6502
4. Zilog Z80.

A utilidade de software compatível do 8080A é maior; o custo é baixo, mas sua complexidade de circuito é também a maior. O 8080A, quando descrito como um "computador de um único chip", conta com vários dispositivos de suportes externos. Sua configuração mínima funcional consiste em três chips como mostrado na figura 2.4. Sua estrutura de via do processador central é similar à da figura 2.3, mas quando combinada com 8224 e 8228 emula a arquitetura de via mais desejável descrita na figura 2.2.

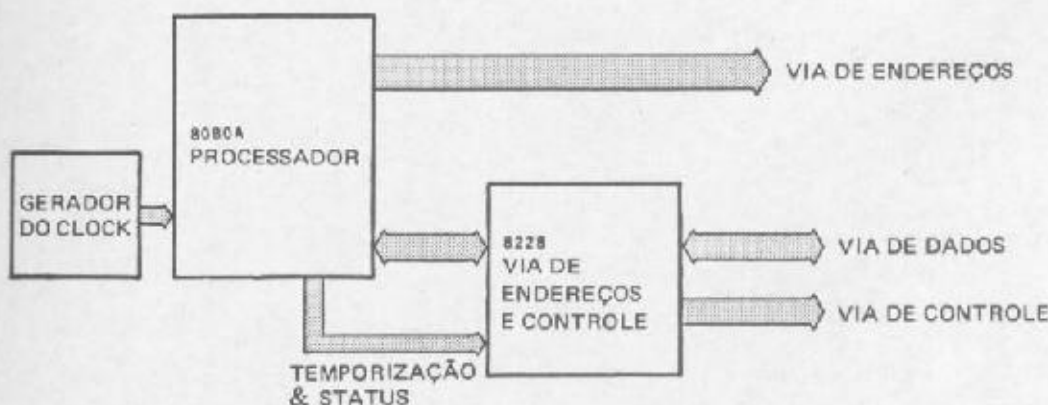



Figura 2.4 Configuração mínima de 8080A, com três chips, ilustrando os dispositivos de suporte necessário. A via de controle contém as funções temporizadas necessárias para decodificar os conteúdos das vias de dados e endereço.

O que há de melhor está incorporado dentro do Z80. Este não somente executa o conjunto completo de instruções do 8080A, como também possui instruções adicionais que servem para fazê-lo um processador muito potente. A estrutura de via do Z80 está ilustrada na figura 2.5. O Z80 é mais caro do que os outros processadores listados. Entretanto, sua necessidade reduzida de circuitos externos resulta em um custo efetivamente comparativa. Assim, a facilidade de interfaces do Z80 torna-o a escolha natural quando da construção de um microcomputador.

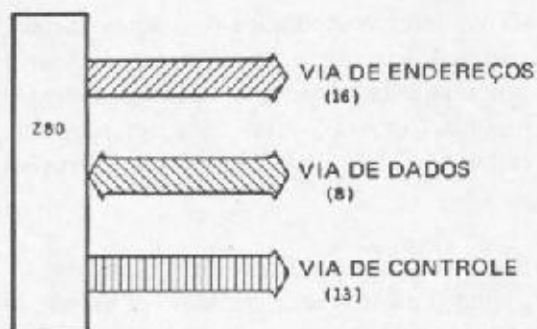



Figura 2.5 Diagrama bloco da estrutura de via do Zilog Z80.

CAPÍTULO 3

O MICROPROCESSADOR Z80

Muito se tem escrito sobre os atributos de *software* e *hardware* do Z80. Embora sem a intenção de imitar outros autores, qualquer livro dedicado à construção de um microcomputador seria incompleto sem uma seção descrevendo o processador. Por meio da completa compreensão da lógica interna e das funções de controle externo do processador central, você será capaz de entender melhor a forma como projetei o restante do sistema de hardware. Você tem muitas opções quando da construção de um computador desde o seu início. Quando mais profundo for seu grau de entendimento, maior será sua confiança, e provavelmente você acrescentará melhorias em seus próprios projetos.

O computador PAZ permite considerável liberdade na seleção de interfaces de periféricos. A escolha depende primeiramente da filosofia de desenvolvimento do sistema, a qual começa com o processador central.

ARQUITETURA DO PROCESSADOR CENTRAL

O Z80 é um microprocessador de registro orientado. Dezoito registros de 8 bits e quatro de 16 bits dentro do processador central são acessíveis ao programador e funcionam como memória programável estática. Esses registros são divididos em dois, principal e alterável, cada um dos quais contém oito registros de 8 bits, de propósito geral que podem ser usados individualmente ou como três pares de registros de 16 bits. Também estão incluídos dois conjuntos de registros acumuladores e de flag. A figura 3.1 ilustra a arquitetura interna do processador central Z80. A figura 3.2 mostra que dentro do Z80 existem registros acumuladores e de flag, com registros gerais e de propósitos especiais.

A seguir é dada uma descrição do funcionamento e da estrutura da maioria dos componentes do processador central.

I. Registros

A. Registros acumulador e de flag

O processador central possui dois pares de registros independentes de acumulador e flag, um no registro principal e o outro no alterável. O acumulador recebe os resultados de todas as operações lógicas e aritméticas de 8 bits, enquanto o registro de flag indica a ocorrência de condições específicas, lógica ou aritmética, no processador, tais como paridade, zero, sinal, carry (transporte), over flow (transbordo). Uma simples troca de instrução permite ao programador selecionar o par de registro de flag ou de acumulador.

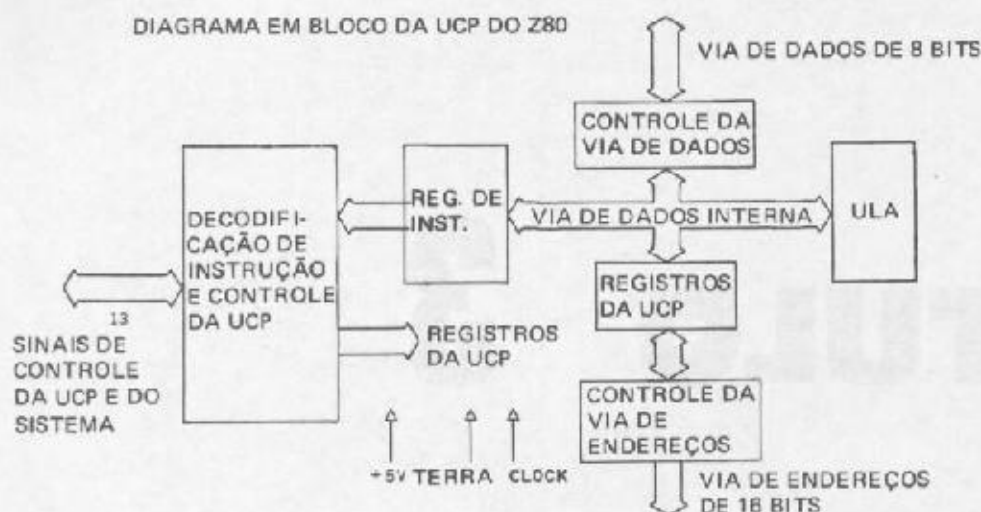



Figura 3.1 Diagrama bloco da arquitetura interna do processador central Z80.

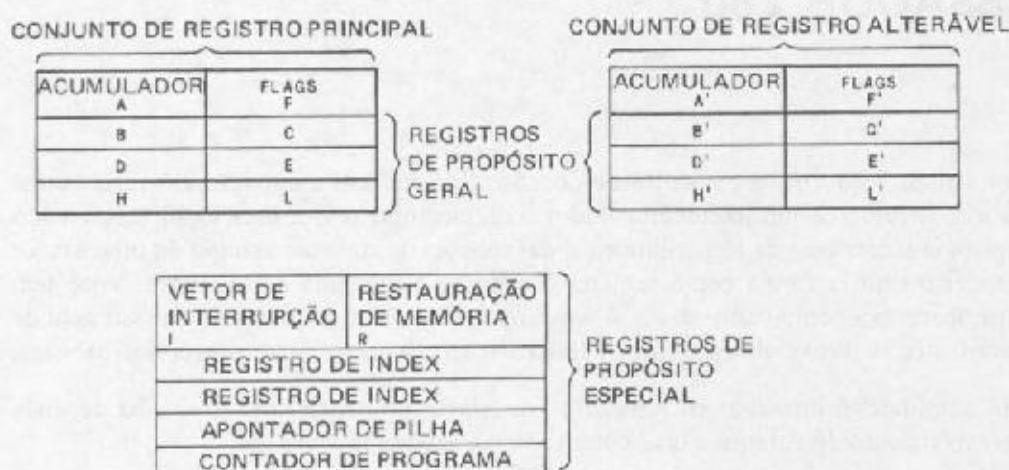



Figura 3.2 Configuração de registro do processador central Z80.

B. Registros de propósito geral

Existem dois registros similares de propósito geral. O registrador principal contém seis registros de 8 bits chamados B, C, D, E, H e L; o registrador alternado também contém seis registros de 8 bits referenciados como B′, C′, D′, E′, H′ e L′. Para operações de 16 bits, esses registros podem ser agrupados em pares de 16 bits (BC, DE, HL ou BC′, DE′, HL′). Uma simples troca de instrução permite ao programador escolher entre os pares de registros.

C. Registros de propósito especial

1. PC (contador de programa)

O contador de programa possui um endereço de memória de 16 bits do qual a instrução em curso será buscada. Seguindo a execução da instrução, o contador PC é incrementado, caso o programa necessite do próximo byte na memória, ou então o conteúdo atual do PC é colocado com um novo valor, se uma instrução de jump ou call for executada.

2. SP (apontador de pilha)

O Z80 permite vários níveis de sub-rotinas acomodando-as através do uso de uma "pilha" e um "apontador de pilha"; quando determinadas instruções são executadas, ou quando chamadas para sub-rotinas são feitas, o contador PC e outro dado pertinente podem ser temporariamente armazenados

na pilha. Uma pilha é uma área reservada de várias posições de memória, o topo da qual é indicado pelo conteúdo do apontador de pilha, o que quer dizer que o apontador de pilha mostra o endereço da entrada mais recente, porque as posições de memória são organizadas como um arquivo do tipo último a entrar, primeiro a sair. Independentemente do tamanho da sub-rotina, ao término desta o processador central retorna ao programa principal através do endereço deixado no topo da pilha. Teoricamente, a pilha pode ter até 64 K bytes; entretanto, o espaço de programa não deve ser sobreescrito por uma pilha extensa.

D. Registros de indexação IX e IY

Esses registros facilitam a manipulação de tabelas de dados. Eles são dois registradores independentes de 16 bits que transformam a base de endereços usada em modos de endereçamento indexado, e aponta para as posições da memória onde o dado pertinente deve ser armazenado ou restaurado. Incorporado nas instruções de indexação está um inteiro, complementado a dois, que especifica o deslocamento desse endereço básico.

E. Registro de endereço de página de interrupção (I)

Este é um registro de 8 bits que pode ser carregado com um endereço da página de uma rotina de interrupção. Um programa de controle de interrupção irá vetoriar esse endereço de página.

F. Registro de restauração de memória (R)

A fim de permitir o uso de memórias dinâmicas para o Z80, um registro de 7 bits para restauração de memória é automaticamente incrementado após cada instrução de busca (fetch).

II. Unidade lógica e aritmética

Manipulações aritméticas e operações lógicas manuseiam 8 bits de cada vez na ULA (Unidade Lógica e Aritmética) do Z80. A ULA comunica-se internamente com os registros do processador central e não é acessível diretamente pelo programador. A ULA executa as seguintes operações:

Deslocamento à direita e à esquerda
Incremento
Decremento
Soma
Subtração
E
Ou
Ou Exclusivo
Comparação
Seta Bit (liga Bit)
Reseta Bit (desliga Bit)
Testa Bit

III. Registro de instrução e controle do processador central

O registro de instrução possui o conteúdo da posição de memória endereçada pelo PC (Contador de Programa) e é carregado durante o ciclo de busca de cada instrução. A unidade de controle do processador central executa as funções definidas pela instrução do registro de instrução e gera todos os sinais de controle necessários para transmitir os resultados aos registros apropriados.

IV. Hardware do processador central

A. A figura 3.3 mostra a pinagem do Z80. Este vem em uma embalagem industrial básica de 40 pinos do tipo dual em linha. A seguir é dada uma lista dos pinos com uma explanação de suas funções:

$A_0 - A_{15}$ (Via de endereços) Saída de três estados, ativa alta. $A_0 - A_{15}$ constitui uma via de 16 bits de endereços. Esses sinais fornecem o endereço para as mudanças de dados na memória (até 64K bytes) e para mudanças de dados nos dispositivos de E/S. O endereçamento de E/S utiliza os oito mais baixos bits de endereço para permitir ao usuário selecionar diretamente até 256 entradas ou 256 portas de saídas. A_0 é o bit de endereço menos significativo. Durante o tempo de restauração os sete bits de mais baixa ordem possuem um endereço válido de restauração.

$D_0 - D_7$ (Via de dados) Entrada/Saída de três estados, ativa alta. $D_0 - D_7$ constitui uma via de dados bidirecional de 8 bits a qual é usada para trocas de dados com a memória e os dispositivos de E/S.

$\overline{M_1}$ (Ciclo 1 de máquina) Saída, ativa baixa. $\overline{M_1}$ indica que o ciclo de máquina presente é o ciclo de busca de uma execução de instrução. Note que durante a execução de 2 bytes, $\overline{M_1}$ é gerado à medida que cada byte é buscado. A codificação desses, 2 bytes sempre começa com CBH, DDH, EDH, ou FDH. $\overline{M_1}$ também ocorre com $\overline{IORQ}$ para indicar o reconhecimento de um ciclo de interrupção.

$\overline{MREQ}$ (Pedido de memória) Saída de três estados, ativa baixa. O sinal de pedido de memória indica que a via de endereço possui um endereço válido para uma operação de leitura de memória ou escrita na memória.

$\overline{IORQ}$ (Pedido de entrada/saída) Saída de três estados, ativa baixa. O sinal $\overline{IORQ}$ indica que a metade menos significativa da via de endereço possui um endereço de E/S válido para uma operação de leitura ou escrita em uma E/S. Um sinal $\overline{IORQ}$ também é gerado com um sinal $\overline{M1}$, quando uma interrupção está sendo reconhecida, para indicar que um vetor de resposta de interrupção pode ser colocado na via de dados. Operações de reconhecimento de interrupção podem ocorrer durante um tempo de $\overline{M1}$ enquanto operações de E/S são proibidas.

$\overline{RD}$ (Leitura de memória) Saída de três estados, ativa baixa. $\overline{RD}$ indica que o processador central quer ler da memória ou de um dispositivo de E/S. O dispositivo de E/S endereçado ou a memória endereçada devem usar este sinal para colocar o dado na via de dados do processador central.

$\overline{WR}$ (Escrita na memória) Saída de três estados, ativa baixa. $\overline{WR}$ indica que a via de dados do processador central possui um dado válido para ser armazenado na memória ou no dispositivo de E/S endereçado.

$\overline{RFSH}$ (Refresh) (Restauração) Saída ativa baixa. $\overline{RFSH}$ indica que os 7 bits de mais baixa ordem da via de endereço contêm um endereço de refresh para memórias dinâmicas e o sinal $\overline{MREQ}$ deve ser usado para fazer uma leitura de refresh para todas as memórias dinâmicas.

$\overline{HALT}$ (Estado parado) Saída, ativa baixa. $\overline{HALT}$ indica que o processador central executou uma instrução HALT e está esperando uma interrupção mascarável ou não mascarável antes de reassumir a operação. Enquanto parado, o processador central executa NOPs (não operação) para manter ativo o refresh (restauração) de memória.

$\overline{WAIT}$ (Espera) Entrada, ativa baixa. O $\overline{WAIT}$ indica para o processador central Z80 que o dispositivo de E/S ou a memória endereçada não estão prontos para transferência de dados. O processador central continua no estado de espera tanto tempo quanto o sinal $\overline{WAIT}$ estiver ativo; este sinal permite a memória ou o dispositivo de E/S serem sincronizados com o processador central.

$\overline{INT}$ (Interrupção) Entrada, ativa baixa. O sinal de pedido de interrupção é gerado pelos dispositivos de E/S. Um pedido será atendido no final da instrução presente se o flip-flop de interrupção controlado por software interno estiver habilitado e se o final $\overline{BUSRQ}$ não estiver ativo. Quando o processador central aceita a interrupção, um sinal de reconhecimento ($\overline{IORQ}$ durante $\overline{M1}$) é enviado para início do próximo ciclo de instrução. O processador central pode responder a uma interrupção de três modos diferentes.

$\overline{\text{NMI}}$ (Interrupção não mascarável) — Entrada, gatilhada (trigger) na descida. A linha de pedido de interrupção não mascarável tem prioridade maior do que o $\overline{\text{INT}}$ e é sempre reconhecida no final da instrução presente, indiferente ao *status* do flip-flop de interrupção. $\overline{\text{NMI}}$ força o processador central a recomeçar da posição 0066_{16}. O contador de programa é automaticamente salvo na pilha externa, dessa forma o usuário pode retornar ao programa que foi interrompido. Note que ciclos contínuos de WAIT podem impedir o término da instrução presente e que um $\overline{\text{BUSRQ}}$ anulará um $\overline{\text{NMI}}$.

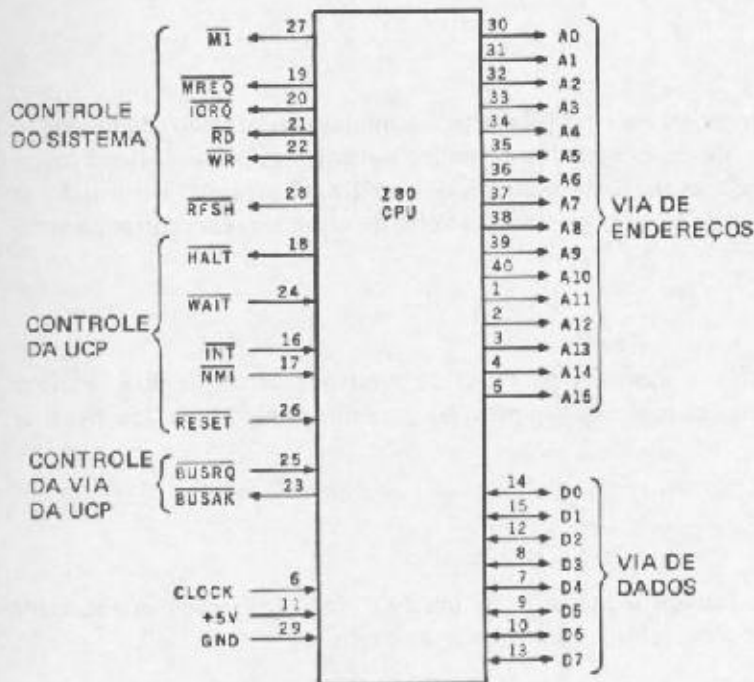



Figura 3.3 Configuração dos pinos para o microprocessador Z80.

A temporização desses sinais será discutida nas seções de hardware.

V. Tipos de instruções do Z80

O Z80 pode executar 158 instruções separadas incluindo todas as 78 do 8080A.

Elas podem ser agrupadas como a seguir:

A. Carga e troca

Instruções de carga movem dados entre registros, ou entre registros e memória. A fonte e destino desses dados são especificados dentro da instrução. Instruções de troca permutam os conteúdos de dois registros.

B. Aritmética e lógica

Essas instruções operam o dado num acumulador, registro ou numa determinada posição de memória. Os resultados são colocados no acumulador e os flags são ligados de acordo. Operações aritméticas incluem adição e subtração de 16 bits entre pares de registros.

C. Transferência de bloco e procura

O Z80 usa uma simples instrução para transferir qualquer tamanho de bloco da memória para qualquer outro grupo contínuo de posições de memória. A busca de bloco usa um simples comando para examinar um bloco de memória através de um caracter particular de 8 bits.

D. Rotação e deslocamento

Dados podem ser rotacionados e deslocados no acumulador, num registro do processador central, ou na memória. Essas instruções também têm facilidades de manuseio do código BCD (binary-coded decimal).

E. Manipulação de bit

A manipulação de bit inclui funções de liga (set), desliga (reset) e teste. Bits individuais podem ser modificados ou testados no acumulador do processador central, ou memória. Os resultados das operações de teste são indicados nos registros de flags.

F. Pulo, chamada e retorno

Um pulo é um desvio para uma localização de programa especificada pelo conteúdo do contador de programa. O conteúdo do contador de programa pode vir de três modos de endereçamento: imediato, estendido, ou registro indireto. Uma chamada é uma forma especial de pulo onde o endereço seguinte à instrução de chamada é colocado na pilha antes do pulo ser feito. Um retorno é a volta da chamada. Essa categoria inclui instruções especiais de reinício.

G. Entrada e saída

Essas instruções transferem dados de registro e memória para um dispositivo externo de E/S. Existem 256 portas de entrada e 256 de saída. Instruções especiais são providas para mover blocos de 256 bytes de ou para portas de E/S e memória.

H. Controle da UCP

Essas instruções incluem parada de UCP ou causam a execução de um NOP (não operação). A faculdade de permitir ou não entrada de interrupção é uma capacidade suplementar de controle.

VI. Formatos de instruções e dados

A memória para o Z80 é organizada em quantidades de 8 bits chamados de bytes (veja figura 3.4). Cada byte de programa é armazenado em uma posição de memória única e é referenciado por um endereço binário de 16 bits.

A capacidade de endereçamento direto da memória é de 65536 bytes (65 K), a qual pode ser formada por qualquer combinação de ROM (read-only memory), EPROM (erasable-programmable read-only memory), ou memória programável. O dado é armazenado nos formatos da figura 3.5.

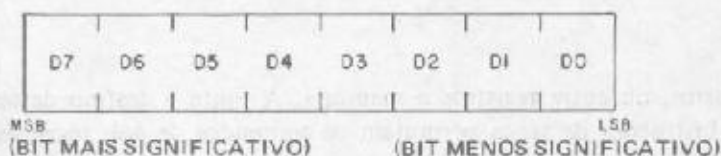



Figura 3.4 Organização de um byte de dado no Z80.

VII. Flags de *status* do Z80

O registro de flag (F e F′) fornece informação para o usuário saber o *status* do processador central em qualquer tempo. Existem quatro bits de flag testáveis e dois não testáveis em cada registro. A figura 3.6 mostra a posição e identidade desses bits de flag.

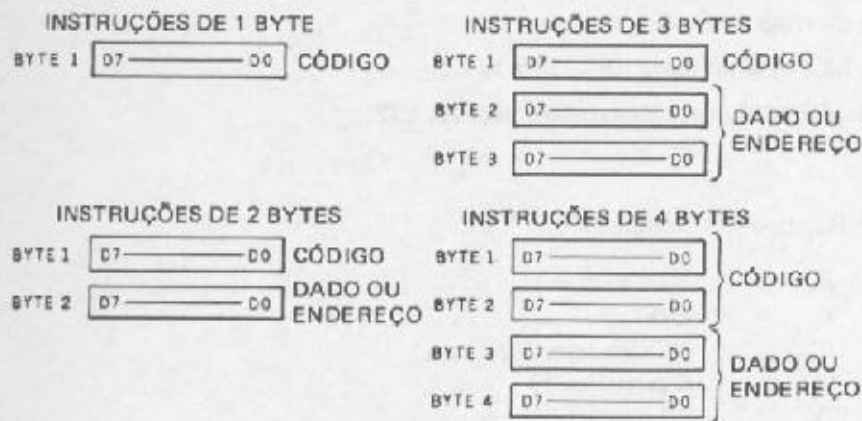



Figura 3.5 Formatos de instrução em linguagem de máquina para o Z80.

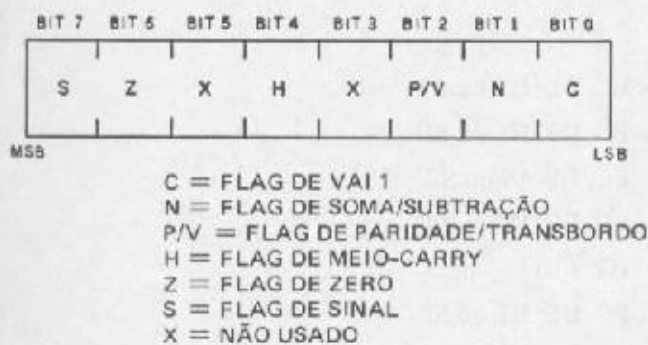



Figura 3.6 Posição e identidade dos bits de *status* de flag no registro de flag.

As instruções setam (bit de flag = 1) ou resetam (bit de flag = 0) os flags de forma relevante para a operação que está sendo executada.

VIII. Conjunto de instruções do Z80

Os seguintes símbolos e abreviações são usados na descrição subseqüente das instruções do Z80:

Símbolo	Significado
acumulador	Registro A
endereço	Uma quantidade de 16 bits de endereço
endereço de ordem alta	Os 8 bits mais significativos do endereço de 16 bits
endereço de ordem baixa	Os 8 bits menos significativos do endereço de 16 bits
dado	Uma quantidade de 8 ou 16 bits
dado de ordem alta	Os 8 bits mais significativos do dado de 16 bits
dado de ordem baixa	Os 8 bits menos significativos do dado de 16 bits
porta	Um endereço de 8 bits de um dispositivo de E/S
r, r'	Um dos registradores A, B, C, D, E, H ou L
n	Uma expressão de 1 byte no range de 0 a 255
nn	Uma expressão de 2 bytes no range de 0 a 65.535
d	Uma expressão de 1 byte no range de −128 a 127

b	Uma expressão no range de 0 a 7
e	Uma expressão de 1 byte no range de −126 a 129
cc	O estado dos flags para as instruções condicionais JR e JP:

cc	Flag Relativo	Condição
000	Z	NZ não zero
001	Z	Z zero
010	C	NC não carry (não vai 1)
011	C	C carry (vai 1)
100	P/V	PO paridade ímpar
101	P/V	PE paridade par
110	S	P sinal positivo
111	S	M sinal negativo

XXH	Indica endereço hexadecimal
qq	Qualquer um dos pares de registros BC, DE, HL ou AF
ss	Qualquer um dos pares de registros BC, DE, HL ou SF
pp	Qualquer um dos pares de registros BC, DE, IX ou SP
rr	Qualquer um dos pares de registros BC, DE, IY, ou SP
s	Qualquer de r, n, (HL), (IX + d) ou (IY + d)
dd	Qualquer um dos pares de registros BC, DE, HL ou SP
m	Qualquer de r, (HL), (IX + d) ou (IY + d)
(HL)	Especifica o conteúdo da memória na posição endereçada pelo conteúdo do par de de registro HL
(nn)	Especifica o conteúdo da memória na posição endereçada pela expressão de 2 bytes em nn
PC	Contador de programa
SP	Apontador de pilha
t	Uma expressão no range de 0 a 7
C, N, P/V, H, Z, S	Flags de condição:

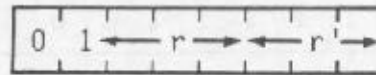
C	Carry (vai 1)
N	Adição/subtração
P/V	Paridade/transbordo
H	Meio carry
Z	Zero
S	Sinal

$\leftarrow$	"é transferido para"
$\wedge$	"E" lógico
$\oplus$	OU exclusivo
$\vee$	OU inclusive
$+$	adição
$-$	subtração
$\leftrightarrow$	"é trocado com"

GRUPO DE CARREGAMENTO DE OITO BITS

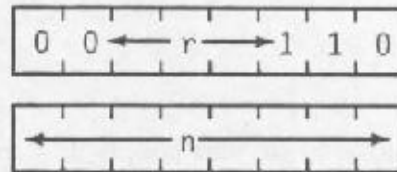
LD r, r'

r ← r'

O conteúdo de qualquer registro r' é carregado em qualquer outro registro r.

Ciclos: 1
Estados: 4
Flags: nenhum

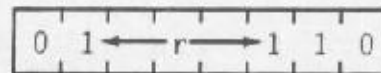
LD r, n

r ← n

O inteiro n de 8 bits é carregado em qualquer registro r.

Ciclos: 2
Estados: 7
Flags: nenhum

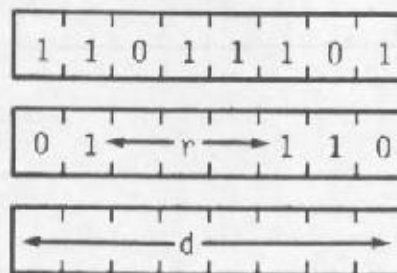
LD r, (HL)

r ← (HL)

O conteúdo de 8 bits da posição de memória (HL) é carregado no registro r.

Ciclos: 2
Estados: 7
Flags: nenhum

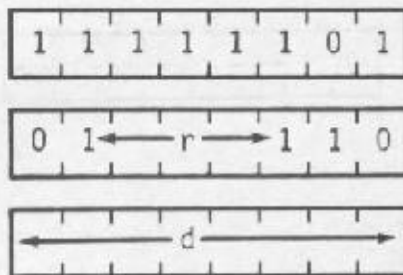
LD r, (IX + d)

r ← (IX + d)

O operando (IX + d) (o conteúdo do registro de index IX somado com um deslocamento inteiro d) é carregado no registro r.

Ciclos: 5
Estados: 19
Flags: nenhum

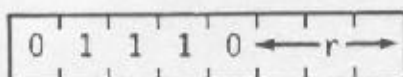
LD r, (IY + d)

r ← (IY + d)

O operando (IY + d) (o conteúdo do registro de index IY somado com um deslocamento inteiro d) é carregado em um registro r.

Ciclos: 5
Estados: 19
Flags: nenhum

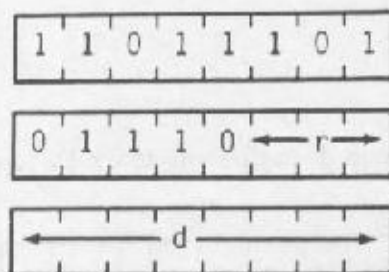
LD (HL), r

(HL) ← r

O conteúdo do registro r é carregado em uma posição de memória especificada pelo conteúdo do par de registro HL.

Ciclos: 2
Estados: 7
Flags: nenhum

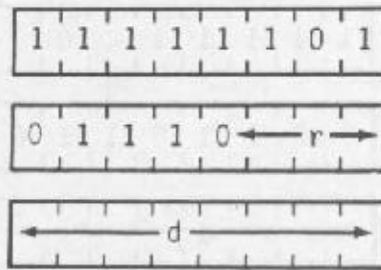
LD (IX + d), r

(IX + d) ← r

O conteúdo do registro r é carregado no endereço de memória especificado pelo conteúdo do registro de index IX somado com d, o qual é um complemento a dois do deslocamento inteiro.

Ciclos: 5
Estados: 19
Flags: nenhum

LD (IY + d), r

(IY + d) ← r

O conteúdo do registro r é carregado no endereço de memória especificado pela soma do conteúdo do registro de index IX com d, um complemento a dois do deslocamento inteiro.

Ciclos: 5
Estados: 19
Flags: nenhum

LD (HL), n

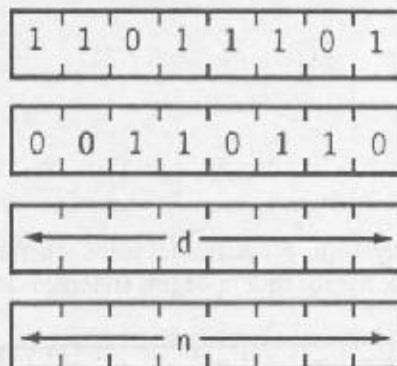
(HL) ← n

O inteiro n é carregado no endereço de memória especificado pelo conteúdo do par de registros HL.

Ciclos: 3
Estados: 10
Flags: nenhum

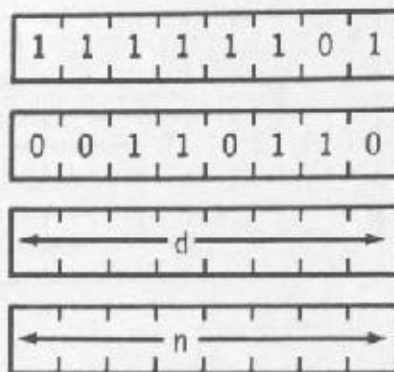
LD (IX + d), n

(IX + d) ← n

O operando n é carregado em um endereço de memória especificado pela soma do conteúdo do registro de index IX e o complemento a dois do deslocamento do operando d.

Ciclos: 5
Estados: 19
Flags: nenhum

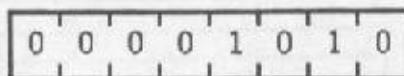
LD (IY + d), n

(IY + d) ← n

O inteiro n é carregado em uma posição de memória especificada pelo conteúdo do registro de index IY somado com um deslocamento inteiro d.

Ciclos: 5
Estados: 19
Flags: nenhum

LD A, (BC)

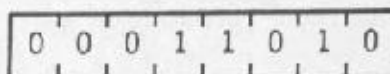
A ← (BC)

O conteúdo da posição de memória especificada pelo conteúdo do par de registro BC é carregado no Acumulador.

Ciclos: 2
Estados: 7
Flags: nenhum

LD A, (DE)

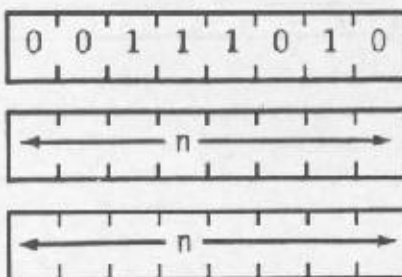
A ← (DE)

O conteúdo da posição de memória especificada pelo par de registro DE é carregado no Acumulador.

0 0 0 1 1 0 1 0

Ciclos: 2
Estados: 7
Flags: nenhum

LD A, (nn)

A ← (nn)

O conteúdo da posição de memória especificada pelos operandos nn é carregado no Acumulador. O primeiro operando n é o byte de menor ordem de um endereço de memória de 2 bytes.

Ciclos: 4
Estados: 13
Flags: nenhum

LD (BC), A

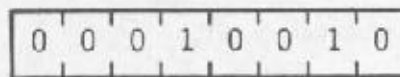
(BC) ← A

O conteúdo do Acumulador é carregado na posição de memória especificada pelo conteúdo do par de registro BC.

Ciclos: 2
Estados: 7
Flags: nenhum

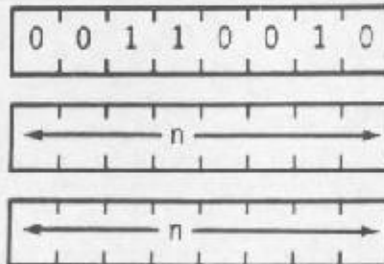
LD (DE), A

(DE) ← A

O conteúdo do Acumulador é carregado na posição de memória especificada pelo par de registro DE.

Ciclos: 2
Estados: 7
Flags: nenhum

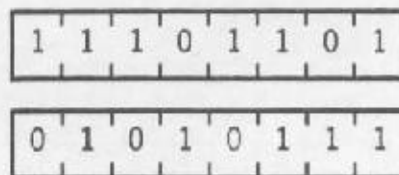
LD (nn), A

(nn) ← A

O conteúdo do Acumulador é carregado no endereço de memória especificado pelos operandos nn. O primeiro operando n é o byte de menor ordem do operando nn.

Ciclos: 4
Estados: 13
Flags: nenhum

LD A, I

A ← I

O conteúdo do registro de controle do vetor de interrupção I é carregado no acumulador.

Ciclos: 2
Estados: 9
Flags: S, Z, M, N, P/V
S: ligado se I < 0
Z: ligado se I = 0
H, N: desligado se I = 0
P/V: contém o conteúdo de IFF2

LD A, R

A ← R

O conteúdo do registro de Refresh de memória R é carregado no Acumulador.

Ciclos: 2
Estados: 9
Flags: S, Z, H, N, P/V
S: ligado se R < 0
Z: ligado se R = 0
H, N: desligado
P/V: contém o conteúdo de IFF2

LD I, A

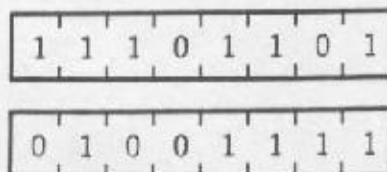
I ← A

O conteúdo do Acumulador é carregado no registro de controle do vetor de interrupção I.

Ciclos: 2
Estados: 9
Flags: nenhum

LD R, A

R ← A

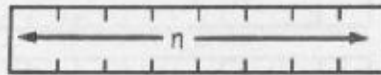
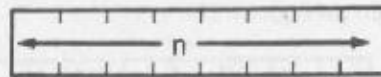
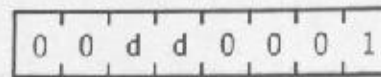
O conteúdo do Acumulador é carregado no registro de Refresh de memória R.

Ciclos: 2
Estados: 9
Flags: nenhum

INSTRUÇÕES DE CARREGAMENTO DE DEZESSEIS BITS

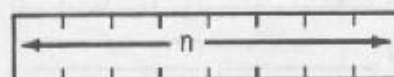
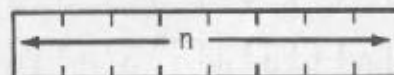
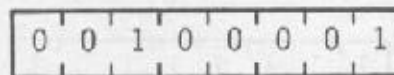
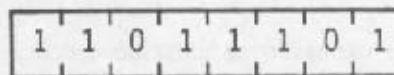
LD dd, nn

dd ← nn

O inteiro nn de 2 bytes é carregado no par de registro dd, onde dd define os pares de registros BC, DE, HL, ou SP, montado no código objeto como a seguir:

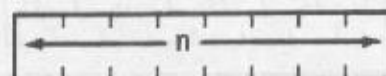
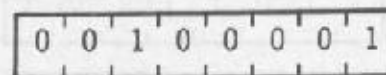
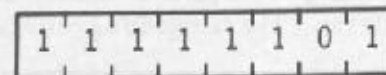
Par	dd
BC	00
DE	01
HL	10
SP	11

Ciclos: 3
Estados: 10
Flags: nenhum

LD IX, nn

IX ← nn
O inteiro nn é carregado no registro de index IX.

Ciclos: 4
Estados: 14
Flags: nenhum

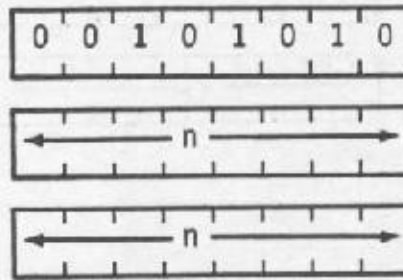
LD IY, nn

IY ← nn
O inteiro nn é carregado no registro de index IY.

1 1 1 1 1 1 0 1
0 0 1 0 0 0 0 1
n
n

Ciclos: 4
Estados: 14
Flags: nenhum

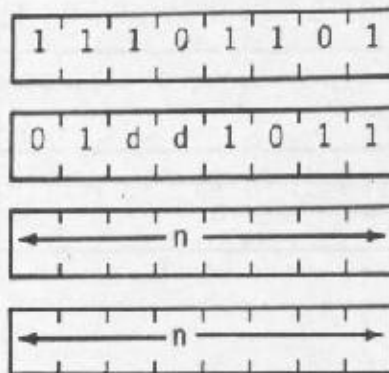
LD HL, (nn)

$H \leftarrow (nn + 1), L \leftarrow (nn)$

O conteúdo do endereço de memória nn é carregado no registro L, e o conteúdo da próxima posição de memória (nn + 1) é carregado no registro H.

Ciclos: 5
Estados: 16
Flags: nenhum

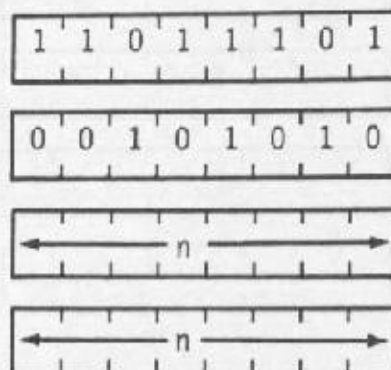
LD dd, (nn)

$dd_H \leftarrow (nn + 1), dd_L \leftarrow (nn)$

O conteúdo do endereço nn é carregado na parte de ordem baixa do par de registro dd, e o conteúdo do próximo endereço de memória (nn + 1) é carregado na parte alta de dd.

Ciclos: 6
Estados: 20
Flags: nenhum

LD IX, (nn)

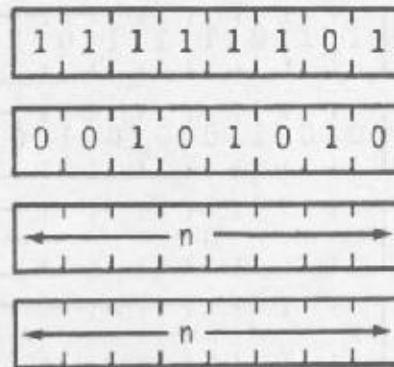
$IX_H \leftarrow (nn + 1), IX_L \leftarrow (nn)$

O conteúdo do endereço nn é carregado na parte de ordem baixa do registro de index IX, e o conteúdo do próximo endereço de memória (nn + 1) é carregado na parte de ordem alta de IX.

Ciclos: 6
Estados: 20
Flags: nenhum

LD IY, (nn)

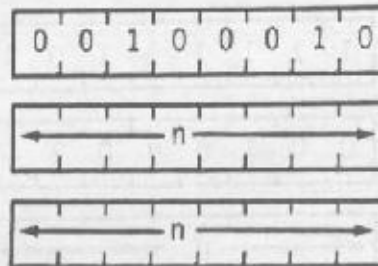
$IY_H \leftarrow (nn+1), IY_L \leftarrow (nn)$

O conteúdo do endereço nn é carregado na parte de ordem baixa do registro de index IY, e o conteúdo do próximo endereço de memória (nn + 1) é carregado na parte de ordem alta de IY.

1 1 1 1 1 1 0 1
0 0 1 0 1 0 1 0
← n →
← n →

Ciclos: 6
Estados: 20
Flags: nenhum

LD (nn), HL

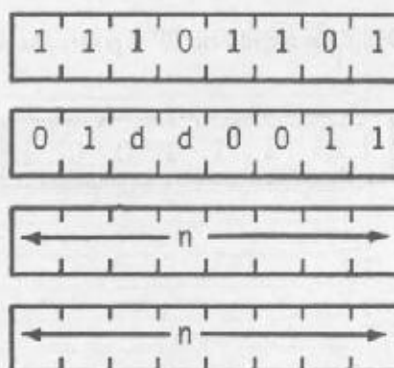
$(nn+1) \leftarrow H, (nn) \leftarrow L$

O conteúdo do registro L é carregado no endereço de memória nn, e o conteúdo do registro H é carregado no próximo endereço (nn + 1).

Ciclos: 5
Estados: 16
Flags: nenhum

LD (nn), dd

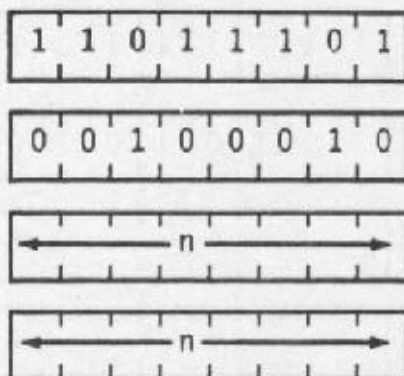
$(nn+1) \leftarrow dd_H, (nn) \leftarrow dd_L$

O byte de baixa ordem do par de registro dd é carregado no endereço de memória nn; o byte superior é carregado no endereço de memória nn + 1.

1 1 1 0 1 1 0 1
0 1 d d 0 0 1 1
← n →
← n →

Ciclos: 6
Estados: 20
Flags: nenhum

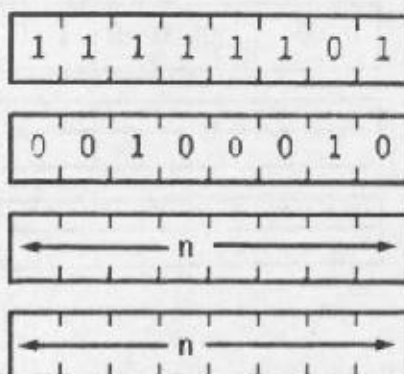
LD (nn), IX

$(nn + 1) \leftarrow IX_H, (nn) \leftarrow IX_L$

O byte de baixa ordem do registro de index IX é carregado no endereço de memória nn; o byte de ordem superior é carregado no próximo endereço nn + 1.

Ciclos: 6
Estados: 20
Flags: nenhum

LD (nn), IY

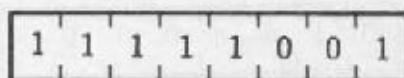
$(nn + 1) \leftarrow IY_H, (nn) \leftarrow IY_L$

O byte de ordem baixa do registro de index IY é carregado no endereço de memória nn; o byte de ordem superior é carregado na posição de memória nn + 1.

Ciclos: 6
Estados: 20
Flags: nenhum

LD SP, HL

$SP \leftarrow HL$

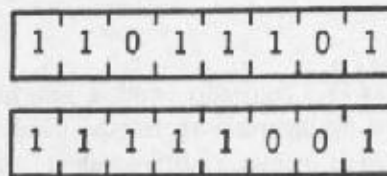
O conteúdo do par de registro HL é carregado no SP (apontador de pilha)

1	1	1	1	1	0	0	1

Ciclos: 1
Estados: 6
Flags: nenhum

LD SP, IX

$SP \leftarrow IX$

O conteúdo de 2 bytes do registro de index IX é carregado no SP.

1	1	0	1	1	1	0	1
1	1	1	1	1	0	0	1

Ciclos: 2
Estados: 10
Flags: nenhum

LD SP, IY

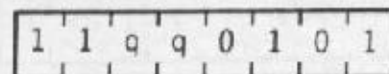
SP ← IY

O conteúdo de 2 bytes do registro de index IY é carregado no SP.

Ciclos: 2
Estados: 10
Flags: nenhum

PUSH qq

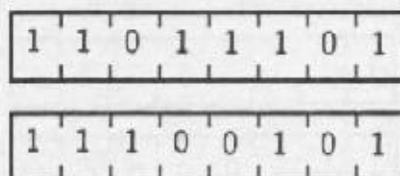
$(SP-2) \leftarrow qq_L, (SP-1) \leftarrow qq_H$

O conteúdo do par de registro qq é colocado na pilha externa de memória LIFO (último a entrar, primeiro a sair). O par de registro SP contém o endereço de 16 bits do topo da pilha. Essa instrução primeiro decrementa o SP e carrega o byte de ordem alta do par de registro qq no endereço de memória agora especificado pelo SP; então decrementa de novo o SP e carrega o byte de ordem baixa do qq na posição de memória correspondente a esse novo endereço no SP.

Ciclos: 3
Estados: 11
Flags: nenhum

PUSH IX

$(SP-2) \leftarrow IX_L$, $(SP-1) \leftarrow IX_H$

O conteúdo do registro de index IX é colocado na pilha. Essa instrução primeiro decrementa o SP e carrega o byte alto de IX no endereço de memória agora especificado pelo SP; então decrementa de novo o SP e carrega o byte baixo na posição de memória correspondente a esse novo endereço no SP.

1	1	0	1	1	1	0	1
1	1	1	0	0	1	0	1

Ciclos: 3
Estados: 15
Flags: nenhum

PUSH IY

$(SP-2) \leftarrow IY_L$, $(SP-1) \leftarrow IY_H$

O conteúdo do registro de index IY é colocado na pilha. Essa instrução primeiro decrementa o SP e carrega o byte alto de IY no endereço de memória agora especificado pelo SP; então decrementa de novo o SP e carrega o byte baixo na posição de memória correspondente a esse novo endereço no SP.

Ciclos: 4
Estados: 15
Flags: nenhum

POP qq

$qq_H \leftarrow (SP+1)$, $qq_L \leftarrow (SP)$

Os 2 bytes do topo da pilha são colocados no par de registro qq. Essa instrução primeiro carrega na parte baixa de qq o byte da posição de memória correspondente ao conteúdo de SP; então SP é incrementado e o conteúdo da posição de memória adjacente é carregado na parte alta de qq, e o SP é agora novamente incrementado.

1	1	q	q	0	0	0	1

Ciclos: 3
Estados: 10
Flags: nenhum

POP IX

$IX_H \leftarrow (SP+1)$, $IX_L \leftarrow (SP)$

Os 2 bytes do topo da pilha são colocados no registro de index IX. Essa instrução primeiro carrega na parte baixa de IX o byte da posição de memória correspondente ao conteúdo de SP; o SP é incrementado e o conteúdo da posição de memória adjacente é carregado na parte alta de IX. O SP é agora novamente incrementado.

Ciclos: 4
Estados: 14
Flags: nenhum

POP IY

$IY_H \leftarrow (SP+1)$, $IY_L \leftarrow (SP)$

Os 2 bytes do topo da pilha são colocados no registro de index IY. Essa instrução primeiro carrega na parte baixa de IY o byte da posição de memória correspondente ao conteúdo de SP; então SP é incrementado e o conteúdo da posição de memória adjacente é carregado na parte alta de IY. O SP é agora novamente incrementado.

1	1	1	1	1	1	0	1

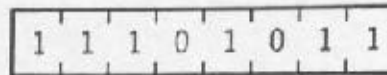
1	1	1	0	0	0	0	1

Ciclos: 4
Estados: 14
Flags: nenhum

GRUPO DE PROCURA, TROCA E TRANSFERÊNCIA DE BLOCO

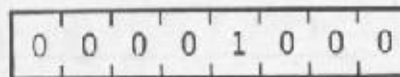
EX DE, HL

DE ⟷ HL

O conteúdo de 2 bytes dos pares de registros DE e HL são trocados.

Ciclos: 1
Estados: 4
Flags: nenhum

EX AF, AF′

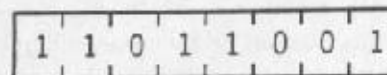
AF ⟷ AF′

O conteúdo de 2 bytes dos pares de registros AF e AF′ são trocados.

Ciclos: 1
Estados: 4
Flags: nenhum

EXX

(BC) ⟷ (BC′), (DE) ⟷ (DE′), (HL) ⟷ (HL′)

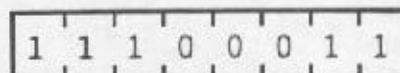
O valor de cada 2 bytes nos pares de registros BC, DE e HL é trocado com valor de 2 bytes no BC′, DE′ e HL′ respectivamente.

1 1 0 1 1 0 0 1

Ciclos: 1
Estados: 4
Flags: nenhum

EX (SP), HL

H ⟷ (SP + 1), L ⟷ (SP)

O byte mais baixo contido no par de registro HL é trocado com o conteúdo do endereço de memória especificado pelo conteúdo do par de registro SP, e o byte mais alto de HL é trocado com o endereço de memória seguinte (SP + 1).

1 1 1 0 0 0 1 1

Ciclos: 5
Estados: 19
Flags: nenhum

EX (SP), IX

IX_H ⟷ (SP + 1), IX_L ⟷ (SP)

O byte mais baixo do registro de index IX é trocado com o conteúdo do endereço de memória especificado pelo conteúdo do par de registro SP, e o byte mais alto de IX é trocado com o próximo endereço (SP + 1).

1	1	0	1	1	1	0	1

1	1	1	0	0	0	1	1

Ciclos: 6
Estados: 23
Flags: nenhum

EX (SP), IY

$IY_H \leftrightarrow (SP+1)$, $IY_L \leftrightarrow (SP)$

O byte mais baixo do registro de index IY é trocado com o conteúdo do endereço de memória especificado pelo conteúdo do par de registro SP, e o byte mais alto de IY é trocado com o próximo endereço de memória.

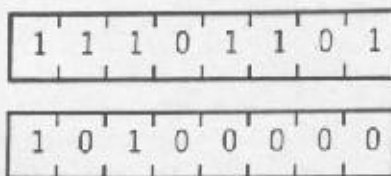
1	1	1	1	1	1	0	1

1	1	1	0	0	0	1	1

Ciclos: 6
Estados: 23
Flags: nenhum

LDI

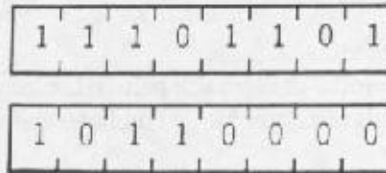
$(DE) \leftarrow (HL)$, $DE \leftarrow DE + 1$, $HL \leftarrow HL + 1$, $BC \leftarrow BC - 1$

Um byte de dado é transferido da posição de memória endereçado pelo conteúdo do par de registro HL para posição de memória endereçada pelo conteúdo do par de registro DE. Então ambos os pares de registros são incrementados e o par de registro BC (contador de byte) é decrementado.

Ciclos: 4
Estados: 16
Flags: H, N, P/V
H, N: desligado
P/V: ligado se $BC - 1 \neq 0$

LDIR

$(DE) \leftarrow (HL)$, $DE \leftarrow DE + 1$, $HL \leftarrow HL + 1$, $BC \leftarrow BC - 1$

Essa instrução de 2 bytes transfere um byte de dado da posição de memória endereçada pelo conteúdo de HL para posição de memória endereçada pelo par de registro DE. Então, ambos os pares de registros são incrementados e o par de registro BC (contador de byte) é decrementado. Se a decrementação fizer com que BC vá a 0, a instrução é terminada. Se BC não é 0, o contador de programa é decrementado de 2 e a instrução é repetida. Nota: se BC for posto em 0 antes da execução da instrução, a instrução será repetida até um total de 64 K bytes. Também, interrupções serão reconhecidas após cada transferência de dado.

1	1	1	0	1	1	0	1
1	0	1	1	0	0	0	0

Para BC ≠ 0:

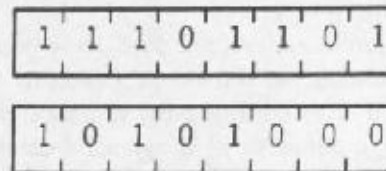
Ciclos: 5
Estados: 21

Para BC = 0

Ciclos: 4
Estados: 16
Flags: H, N, P/V: desligado

LDD

(DE) ← (HL), DE ← DE – 1, HL ← HL – 1, BC ← BC – 1
Essa instrução de 2 bytes transfere um byte da posição de memória endereçada pelo conteúdo de HL para posição de memória endereçada pelo conteúdo de DE. Então ambos os registros incluindo o BC são decrementados.

1	1	1	0	1	1	0	1
1	0	1	0	1	0	0	0

Ciclos: 4
Estados: 16
Flags: H, N, P/V
H, N: desligado
P/V: ligado se BC – 1 ≠ 0

LDDR

(DE) ← (HL), DE ← DE – 1, HL ← HL – 1, BC ← BC – 1
Essa instrução de 2 bytes transfere um byte de dado da posição de memória endereçada pelo conteúdo de HL para posição de memória endereçada pelo conteúdo de DE, Então ambos os registros, tanto quanto o BC, são decrementados. Se a decrementação fizer com que BC vá para 0, a instrução é terminada. Se BC não é 0, o contador de programa é decrementado de 2 e a instrução é repetida. Nota: Se BC é feito 0 antes da execução da instrução, a instrução será repetida até o total de 64 K bytes. Também serão reconhecidas interrupções após cada transferência de dados.

1	1	1	0	1	1	0	1
1	0	1	1	1	0	0	0

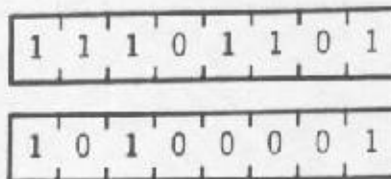
Para BC ≠ 0

Ciclos: 5
Estados: 21

Para BC = 0:

Ciclos: 4
Estados: 16
Flags: H, N, P/V: desligado

CPI

A – (HL), HL ← HL + 1, BC ← BC – 1
O conteúdo da posição de memória endereçada pelo HL é comparada com o conteúdo do Acumulador. No caso de uma comparação verdadeira, um bit de condição é ativado.

Ciclos: 4
Estados: 16
Flags: S, Z, H, N, P/V
S: ligado se o resultado é negativo
Z: ligado se A = (HL)
H: ligado se não houver vem 1 do bit 4
N: ligado
P/V: ligado se BC – 1 ≠ 0

CPIR

A – (HL), HL ← HL + 1, BC ← BC – 1
O conteúdo da posição de memória endereçada pelo HL é comparado com o conteúdo do Acumulador. No caso de uma comparação verdadeira, um bit de condição é ativado. O HL é incrementado e o BC é decrementado. Se o decremento fizer com que BC vá para 0 ou se A = (HL), a instrução é terminada. Se BC não é 0 e se A ≠ (HL), o contador de programa é decrementado de dois, e a instrução é repetida. Nota: Se BC é posto em 0 antes da execução da instrução, a instrução será repetida até um total de 64 K bytes se nenhuma igualdade for encontrada. Também, interrupções serão reconhecidas após cada comparação.

Para BC ≠ 0 e A ≠ (HL):

Ciclos: 5
Estados: 21

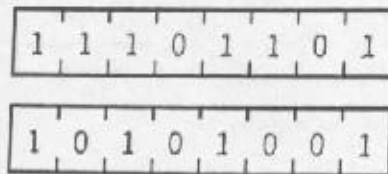
Para BC = ou A = (HL):

Ciclos: 4
Estados: 16
Flags: S, Z, H, N, P/V
S: ligado se o resultado é negativo
Z: ligado se A = (HL);
H: ligado se não houver vem 1 do bit 4
N: ligado
P/V: ligado se BC – ≠ 0

CPD

A – (HL), HL ← HL – 1, BC ← BC – 1
O conteúdo da posição de memória endereçada pelo HL é comparado com o conteúdo do Acumulador. No caso de uma comparação verdadeira um bit de condição é ativado. O HL e o BC são decrementados.

1 1 1 0 1 1 0 1

1 0 1 0 1 0 0 1

Ciclos: 4
Estados: 16
Flags: S, Z, H, N, P/V
S: ligado se o resultado é negativo
Z: ligado se A = (HL)
H: ligado se não houver vem 1 do bit 4
N: ligado
P/V: ligado se BC − 1 ≠ 0

CPDR

A − (HL), HL ← HL − 1, BC ← BC −1

O conteúdo da posição de memória endereçada pelo HL é comparado com o conteúdo do Acumulador. No caso de uma comparação verdadeira um bit de condição é ativado. O HL e o BC são decrementados. Se o decremento fizer com que BC vá para 0 ou se A = (HL), a instrução é terminada. Se BC não é 0 e A ≠ (HL), o contador de programa é decrementado de 2 e a instrução é repetida. Nota: Se BC é colocado em 0 antes da execução da instrução, a instrução fará uma comparação nos 64 K bytes se nenhuma igualdade for encontrada. Também interrupções serão reconhecidas após cada comparação e dado.

Para BC ≠ 0 e A ≠ (HL):

Ciclos: 5
Estados: 21

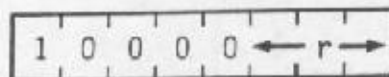
Para BC = 0 ou A = (HL):

Ciclos: 4
Estados: 16
Flags: S, Z, H, N, P/V
S: ligado se o resultado é negativo
Z: ligado se A = (HL)
H: ligado se não houver vem 1 do bit 4
N: ligado
P/V: ligado se BC − 1 ≠ 0

GRUPO LÓGICO E ARITMÉTICO DE 8 BITS

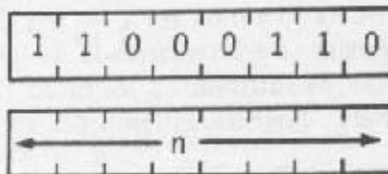
ADDA, r

A ← A + r

O conteúdo do registro r é somado com o conteúdo do Acumulador e o resultado é armazenado no Acumulador

Ciclos: 1
Estados: 4
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado se houver vai 1 do bit 3
N: desligado
C: ligado se houver vai 1 do bit 7
P/V: ligado se houver transbordo

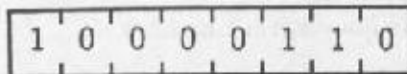
ADDA, n

A ← A + n

O inteiro n é somado com o conteúdo do Acumulador, e o resultado é armazenado no Acumulador.

Ciclos: 2
Estados: 7
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado se houver vai 1 do bit 3
N: desligado
C: ligado se houver vai 1 do bit 7
P/V: ligado se houver transbordo

ADD A, (HL)

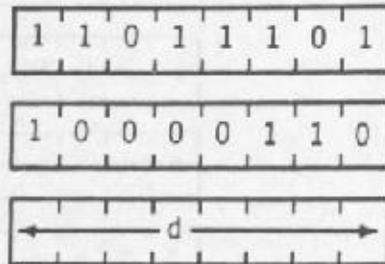
A ← A + (HL)

O byte do endereço de memória especificado pelo conteúdo de HL é somado com o conteúdo do acumulador, e o resultado é armazenado no acumulador.

1 0 0 0 0 1 1 0

Ciclos: 2
Estados: 7
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado se houver vai 1 do bit 3
N: desligado
C: ligado se houver vai 1 do bit 7
P/V: ligado se houver transbordo

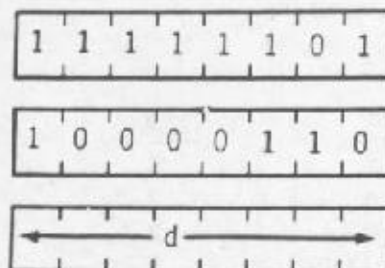
ADD A, (IX + d)

A ← A + (IX + d)

O conteúdo do registro de index IX é somado com um deslocamento d para apontar para um endereço na memória. O conteúdo desse endereço é, então, somado com o conteúdo do Acumulador e o resultado é armazenado no Acumulador.

Ciclos: 5
Estados: 19
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado se houver vai 1 do bit 3
N: desligado
C: ligado se houver vai 1 do bit 7
P/V: ligado se houver transbordo

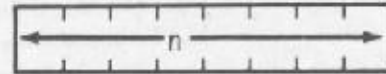
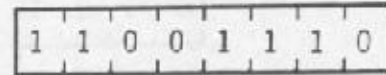
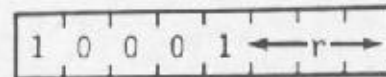
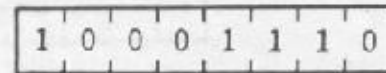
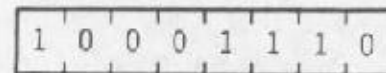
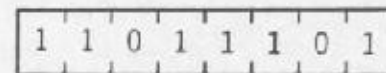
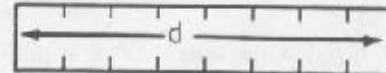
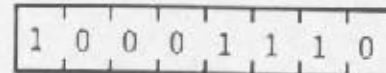
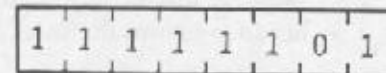
ADD A, (IY + d)

A ← A + (IY + d)

O conteúdo do registro de index IY é somado com um deslocamento d para apontar um endereço na memória. O conteúdo desse endereço é então, somado com o conteúdo do Acumulador, e o resultado é armazenado no Acumulador.

Ciclos: 5
Estados: 19
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado se houver vai 1 do bit 3
N: ligado
C: ligado se houver vai 1 do bit 7
P/V: ligado se houver transbordo

ADC A, s

A ← A + s + CY

O operando s é qualquer de r, n, (HL), (IX + d), ou (IY + d), como definido para instrução análoga ADD. Essas várias possibilidades de combinações de operando são montadas no código objeto da seguinte forma:

Instrução	Código objeto
ADC A, r	1 0 0 0 1 ←r→
ADC A, n	1 1 0 0 1 1 1 0
	←n→
ADC A, (HL)	1 0 0 0 1 1 1 0
ADC A, (IX+d)	1 1 0 1 1 1 0 1
	1 0 0 0 1 1 1 0
	←d→
ADC A, (IY+d)	1 1 1 1 1 1 0 1
	1 0 0 0 1 1 1 0
	←d→

O operando s junto com o Flag vai 1 (carry) ("C" no registro F) é somado com o conteúdo do Acumulador e o resultado é armazenado no Acumulador.

Instrução	*Ciclos*	*Estados*
ADC A, r	1	4
ADC A, n	2	7
ADC A, (HL)	2	7
ADC A, (IX+d)	5	19
ADC A, (IY+d)	5	19

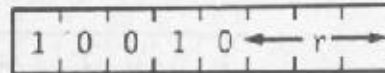
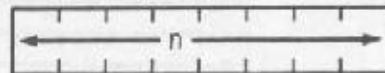
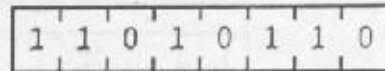
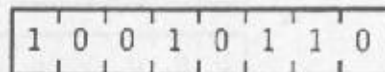
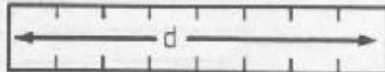
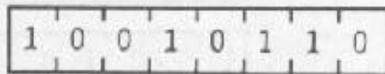
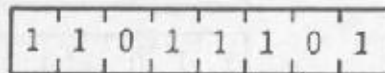
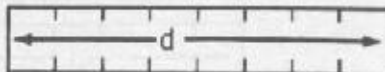
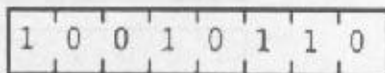
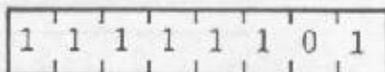
Flags: S, Z, H, N, C, P/V

S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado se houver vai 1 do bit 3
N: desligado
C: ligado se houver vai 1 do bit 7
P/V: ligado se houver transbordo

SUB s

A ← A – s

O operando s é subtraído do Acumulador, e o resultado é armazenado no Acumulador

SUB r	1 0 0 1 0 ←r→
SUB n	1 1 0 1 0 1 1 0
	←n→
SUB (HL)	1 0 0 1 0 1 1 0
SUB (IX+d)	1 1 0 1 1 1 0 1
	1 0 0 1 0 1 1 0
	←d→
SUB (IY+d)	1 1 1 1 1 1 0 1
	1 0 0 1 0 1 1 0
	←d→

Instrução	*Ciclos*	*Estados*
SUB r	1	4
SUB n	2	7
SUB (HL)	2	7
SUB (IX+d)	5	19
SUB (IY+d)	5	19

Flags: S, Z, H, N, C/ P/V

S: ligado se o resultado é negativo

Z: ligado se o resultado é 0

H: ligado se não houver vem 1 do bit 4

N: ligado

C: ligado se não houver vem 1

P/V: ligado se houver transbordo

SBC A, s

A ← A − s − CY

O operando s junto com o Flag vai 1 (carry) ("C" no registro F) é subtraído do conteúdo do Acumulador e o resultado é armazenado no Acumulador.

SBC A, r

1	0	0	1	1	←	r	→

SBC A, n

1	1	0	1	1	1	1	0
←			n				→

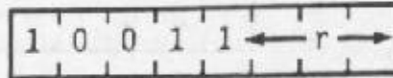
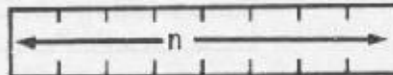
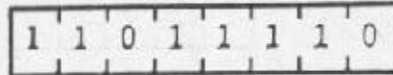
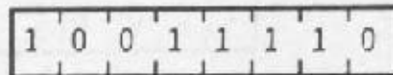
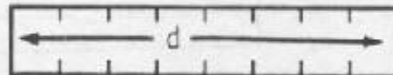
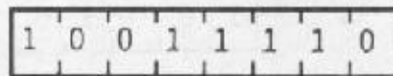
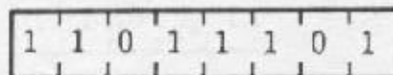
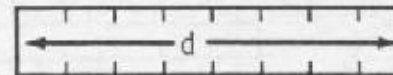
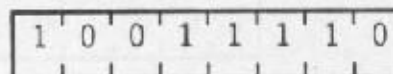
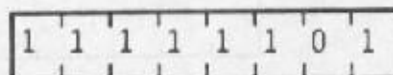
SBC A, (HL)

1	0	0	1	1	1	1	0

SBC A, (IX+d)

1	1	0	1	1	1	0	1
1	0	0	1	1	1	1	0
←			d				→

SBC A, (IY+d)

1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	0
←			d				→

Instrução	*Ciclos*	*Estados*
SBC A, r	1	4
SBC A, n	2	7
SBC A, (HL)	2	7
SBC A, (IX+d)	5	19
SBC A, (IY+d)	5	19

Flags: S, Z, H, N, C, P/V

S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado se não houver vem 1 do bit 4
N: ligado
C: ligado se não houver vai 1
P/V: ligado se houver transbordo

AND s

A ← A ∧ s
Um operando lógico E, bit a bit, é feito entre o byte especificado pelo operando s e o byte contido no Acumulador, o resultado é armazenado no Acumulador.

AND r

1	0	1	0	0	← r →		

AND n

1	1	1	0	0	1	1	0
←			n				→

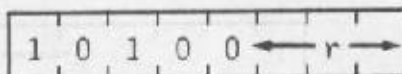
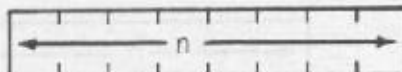
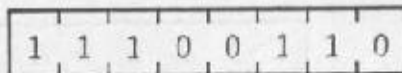
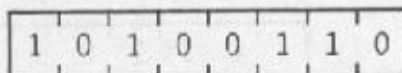
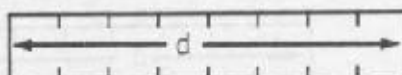
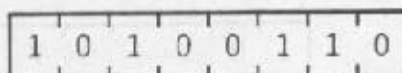
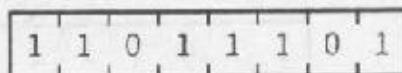
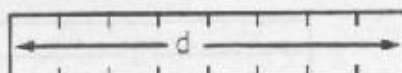
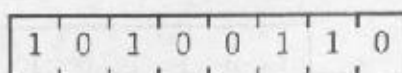
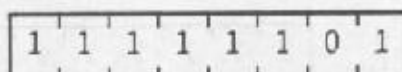
AND (HL)

1	0	1	0	0	1	1	0

AND (IX+d)

1	1	0	1	1	1	0	1
1	0	1	0	0	1	1	0
←			d				→

AND (IY+d)

1	1	1	1	1	1	0	1
1	0	1	0	0	1	1	0
←			d				→

Instrução	*Ciclos*	*Estados*
AND r	1	4
AND n	2	7
AND (HL)	2	7
AND (IX+d)	5	19
AND (IX+d)	5	19

Flags: S, Z, H, N, C, P/V
S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado
N: desligado
C: desligado
P/V: ligado se paridade par

OR s

$A \leftarrow A \vee s$

Uma operação lógica OR(OU), bit a bit, é feita entre o byte especificado pelo operando s e o byte contido no Acumulador; o resultado é armazenado no Acumulador.

OR r

1 0 1 1 0 ← r →

OR n

1 1 1 1 0 1 1 0
← n →

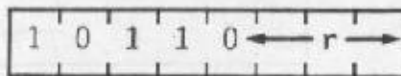
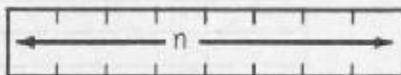
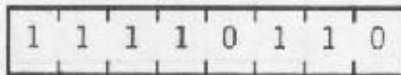
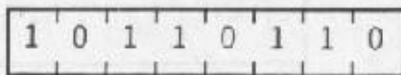
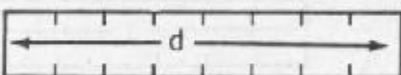
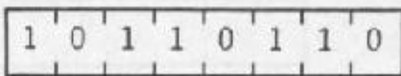
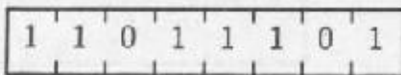
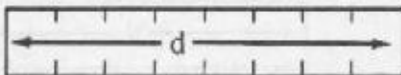
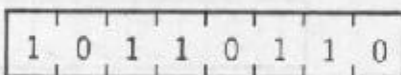
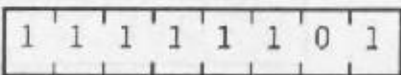
OR(HL)

1 0 1 1 0 1 1 0

OR(IX+d)

1 1 0 1 1 1 0 1
1 0 1 1 0 1 1 0
← d →

OR(IY+d)

1 1 1 1 1 1 0 1
1 0 1 1 0 1 1 0
← d →

Instrução	*Ciclos*	*Estados*
OR r	1	4
OR n	2	7
OR (HL)	2	7
OR (IX+d)	5	19
OR (IY+d)	5	19

Flags: S, Z, H, N, C, P/V

S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado
N: desligado
C: desligado
P/V: ligado se paridade par

XOR s

$A \leftarrow A \oplus s$

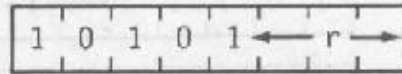
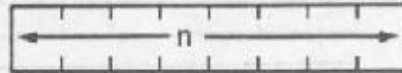
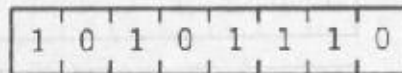
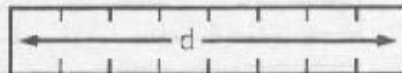
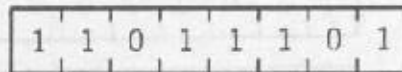
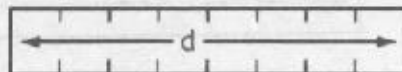
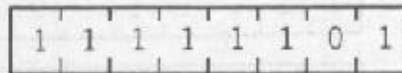
Uma operação lógica OU-exclusivo, bit a bit, é feita entre o byte especificado pelo operando s e o byte contido no Acumulador; o resultado é armazenado no Acumulador.

```
XOR r        1 0 1 0 1 ←-r-→

XOR n        1 1 1 0 1 1 1 0
             ←------n------→

XOR (HL)     1 0 1 0 1 1 1 0

XOR (IX+d)   1 1 0 1 1 1 0 1
             1 0 1 0 1 1 1 0
             ←------d------→

XOR (IY+d)   1 1 1 1 1 1 0 1
             1 0 1 0 1 1 1 0
             ←------d------→
```

Instrução	*Ciclos*	*Estados*
XOR r	1	4
XOR n	2	7
XOR (HL)	2	7
XOR (IX+d)	5	19
XOR (IY+d)	5	19

Flags: S, Z, H, N, C, P/V

S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado
N: desligado
C: desligado
P/V: ligado se paridade par

CP s

A − s
O conteúdo do operando s é comparado com o conteúdo do Acumulador. Se houver uma comparação verdadeira, um flag é ativado.

CP r

1	0	1	1	1	← r →		

CP n

1	1	1	1	1	1	1	0
←			n				→

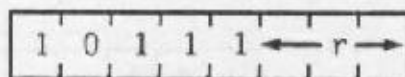
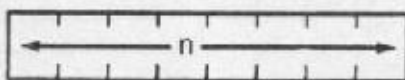
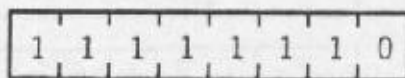
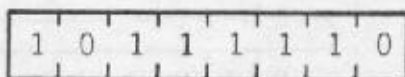
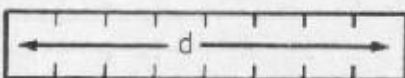
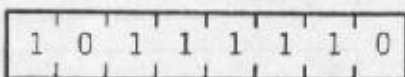
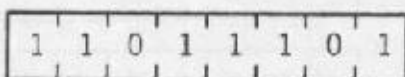
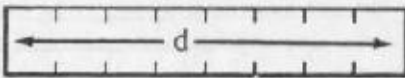
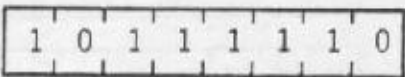
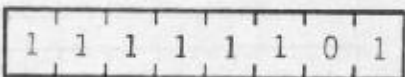
CP (HL)

1	0	1	1	1	1	1	0

CP (IX+d)

1	1	0	1	1	1	0	1
1	0	1	1	1	1	1	0
←			d				→

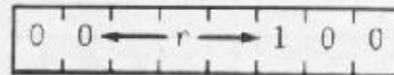
CP (IY+d)

1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	0
←			d				→

Instrução	*Ciclos*	*Estados*
CP r	1	4
CP n	2	7
CP (HL)	2	7
CP (IX+d)	5	19
CP (IY+d)	5	19

Flags: S, Z, H, N, C, P/V
S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado se não houver vem 1 do bit 4
N: ligado
C: ligado se não houver vem 1
P/V: ligado se houver transbordo

INC r

r ← r + 1

O registro r é incrementado

Ciclos: 1

Estados: 4

Flags: S, Z, H, N, P/V

S: ligado se o resultado é negativo

Z: ligado se o resultado é 0

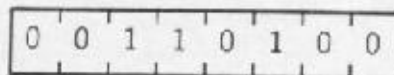
H: ligado se houver vai 1 do bit 3

N: desligado

P/V: ligado se r continha 7FH antes da operação.

INC (HL)

(HL) ← (HL) + 1

O byte contido no endereço especificado pelo conteúdo de HL é incrementado.

Ciclos: 3

Estados: 11

Flags: S, Z, H, N, P/V

S: ligado se o resultado é negativo

Z: ligado se o resultado é 0

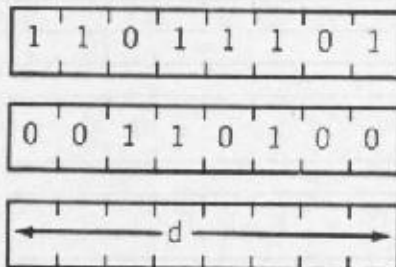
H: ligado se houver vai 1 do bit 3

N: desligado

P/V: ligado se (HL) continha 7FH antes da operação

INC (IX + d)

(IX + d) ← (IX + d) + 1

O conteúdo de index IX é somado com o complemento a dois do deslocamento inteiro d para apontar para um endereço na memória. O conteúdo desse endereço é, então, incrementado.

Ciclos: 6

Estados: 23

Flags: S, Z, H, N, P/V

S: ligado se o resultado é negativo

Z: ligado se o resultado é 0

H: ligado se houver vai 1 do bit 3

N: desligado

P/V: ligado se (IX + d) continha 7FH antes da operação

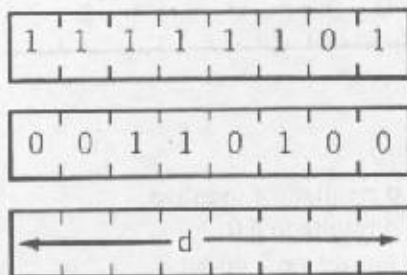
INC (IY + d)

(IY + d) ← (IY + d) + 1

O conteúdo do registro de index IY é somado com o complemento a dois do deslocamento inteiro d para apontar para um endereço na memória. O conteúdo desse endereço é, então, incrementado.

1	1	1	1	1	1	0	1

0	0	1	1	0	1	0	0

← d →

Ciclos: 6
Estados: 23
Flags: S, Z, H, N, P/V

S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado se houver vai 1 do bit 3
N: desligado
P/V: ligado se (IY + d) continha 7FH antes da operação

DEC m

m ← m − 1

O byte especificado pelo operando m é decrementado.

DEC r

0	0	← r →			1	0	1

DEC (HL)

0	0	1	1	0	1	0	1

DEC (IX+d)

1	1	0	1	1	1	0	1

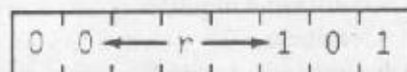
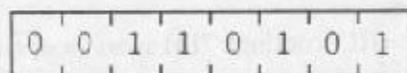
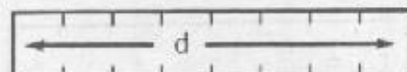
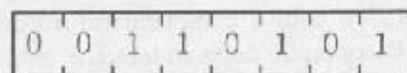
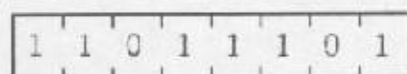
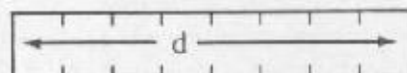
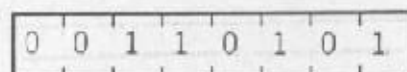
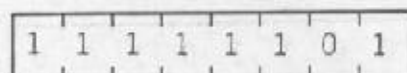
0	0	1	1	0	1	0	1

← d →

DEC (IY+d)

1	1	1	1	1	1	0	1

0	0	1	1	0	1	0	1

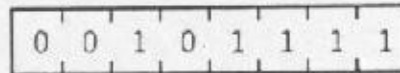
← d →

Instrução	*Ciclos*	*Estados*
DEC r	1	4
DEC (HL)	3	11
DEC (IX+d)	6	23
DEC (IY+d)	6	23

Flags: S, Z, H, N, P/V
S: ligado se o resultado é negativo
Z: ligado se o resultado é 0
H: ligado se não houver vai 1 do bit 4
N: ligado
P/V: ligado se m continha 80 H antes da operação

GRUPOS DE CONTROLE DA CPU E PROPÓSITO GERAL ARITMÉTICO

CPL

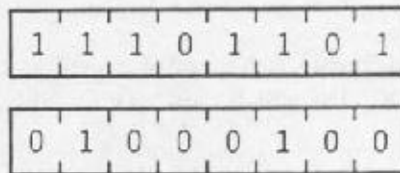
$A \leftarrow \overline{A}$

O conteúdo do Acumulador é invertido (complemento em um)

Ciclos: 1
Estados: 4
Flags: H, N
H: ligado
N: ligado

NEG

$A \rightarrow 0 - A$

O conteúdo do Acumulador é invertido (complemento de dois).

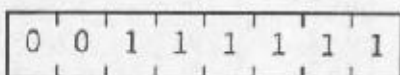
Isto é o mesmo que subtrair o conteúdo do Acumulador de 0 (zero).

1	1	1	0	1	1	0	1
0	1	0	0	0	1	0	0

Ciclos: 2
Estados: 8
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado é negativo
Z: ligado se o resultado é zero
H: ligado se for 1 do bit 4
N: ligado
C: ligado se o Acumulador não era OOH antes da operação
P/V: ligado se o Acumulador era 80 H antes da operação

CCF

$CY \leftarrow \overline{CY}$

O flag C no registrador F é invertido.

0	0	1	1	1	1	1	1

Ciclos: 1
Estados: 4
Flags: H, N, C
H: o CARRY (vai um) antigo será copiado
N: desligado
C: ligado se CY era zero antes da operação

SCF

CY ← 1
O flag C no registrador F é ligado

Ciclos: 1
Estados: 4
Flags: H, N, C
H: desligado
N: desligado
C: ligado

NOP

O processador central não executa nenhuma operação durante este ciclo de máquina.

Ciclos: 1
Estados: 4
Flags: nenhum

DAA ----

Esta instrução ajusta condicionalmente o Acumulador para as operações de soma e subtração em BCD. Para as instruções (ADD, ADC, INC, SUB, SBC, DEC, NEG) a tabela, a seguir, mostra a operação realizada.

OPERAÇÃO	C ANTES DO DAA	VALOR EM HEXA DO DÍGITO SUPERIOR (Bit 7-4)	H ANTES DO DAA	VALOR EM HEXA DO DÍGITO INFERIOR (Bit 3-0)	NÚMERO SOMADO AO BYTE	C DEPOIS DO DAA
ADD ADC INC	0	0–9	0	0–9	00	0
	0	0–8	0	A–F	06	0
	0	0–9	1	0–3	06	0
	0	A–F	0	0–9	60	1
	0	9–F	0	A–F	66	1
	0	A–F	1	0–3	66	1
	1	0–2	0	0–9	60	1
	1	0–2	0	A–F	66	1
	1	0–3	1	0–3	66	1
SUB SBC DEC NEG	0	0–9	0	0–9	00	0
	0	0–8	1	6–F	FA	0
	1	7–F	0	0–9	A0	1
	1	6–F	1	6–F	9A	1

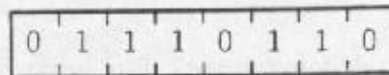
CICLOS M:1 ESTADOS T: 4 4 MHZ

Ciclos: 1
Estados: 4
Flags: S, Z, H, C, P/V
S: ligado se o bit mais significativo do Acumulador for 1 depois da operação
Z: ligado se o Acumulador for zero após a operação
H: veja a instrução
C: veja a instrução
P/V: ligado se o Acumulador é paridade par após a operação

HALT

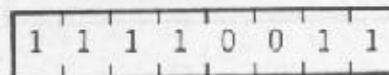
A instrução HALT suspende a operação do processador central até ocorrer uma interrupção ou um rearme do sistema. Enquanto estiver em HALT, o processador estará executando NOP para que seja mantida a restauração da memória.

0 1 1 1 0 1 1 0

Ciclos: 1
Estados: 4
Flags: nenhum

DI

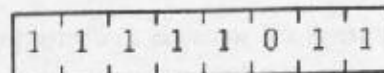
IFF ← 0
DI desabilita a interrupção desligando os FLIP-FLOPS (IFF1 e IFF2). NOTA: Esta instrução desabilita a interrupção durante a sua execução.

1 1 1 1 0 0 1 1

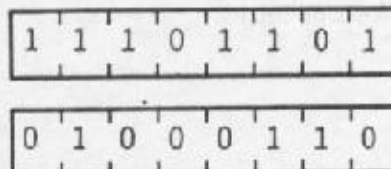
Ciclos: 1
Estados: 4
Flags: nenhum

EI

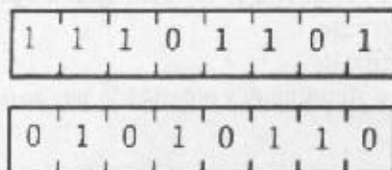
IFF ← 1
EI permite a interrupção ligando os FLIP-FLOPS (IFF1 e IFF2). NOTA: Esta instrução permite a interrupção durante a sua execução.

1 1 1 1 1 0 1 1

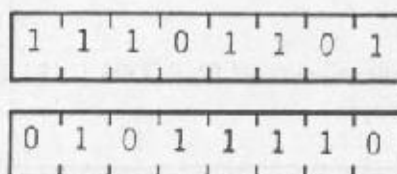
Ciclos: 1
Estados: 4
Flags: nenhum

IM 0

A instrução (IM 0) liga o modo de interrupção 0. Neste modo o dispositivo que estiver interrompendo pode inserir qualquer instrução na via de dados e permitir que o processador central a execute.

Ciclos: 2
Estados: 8
Flags: nenhum

IM 1

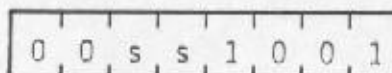
A instrução (IM 1) liga o modelo de interrupção 1. Neste modo o processador responderá uma interrupção executando a instrução que estiver no endereço 0038H.

Ciclos: 2
Estados: 8
Flags: nenhum

IM 2

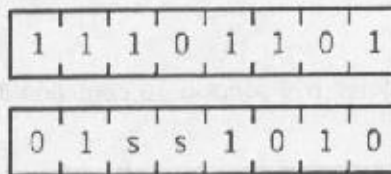
A instrução (IM2) liga o modo de interrupção 2. Este modo permite uma chamada indireta a qualquer posição de memória. Com este modo, o processador monta um endereço de memória de 16 bits. Os 8 bits superiores são o conteúdo do registrador do vetor de interrupção (I) e os 8 bits inferiores são supridos pelo dispositivo que interrompeu.

Ciclos: 2
Estados: 8
Flags: nenhum

GRUPO DE INSTRUÇÕES ARITMÉTICAS DE 16 BITS

ADD HL, ss

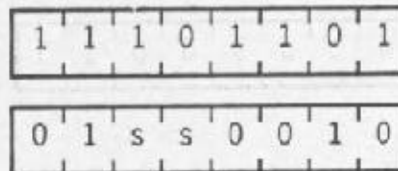
HL ← HL + ss
O conteúdo do par de registradores ss são somados ao conteúdo do par de registradores HL e o resultado é guardado em HL.

Ciclos: 3
Estados: 11
Flags: H, N, C
H: ligado se for 1 além do bit 11
N: desligado
C: ligado se for 1 do bit 15

ADC HL, ss

HL ← HL + ss + CY
O conteúdo do par de registradores ss é somado com flag CARRY (vai um) e com o conteúdo do par de registradores HL, o resultado é guardado em HL.

Ciclos: 4
Estados: 15
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado for negativo
Z: ligado se o resultado for zero
H: ligado se houver CARRY do bit 11
N: desligado
C: ligado se houver CARRY do bit 15
P/V: ligado se houver transbordo (overflow)

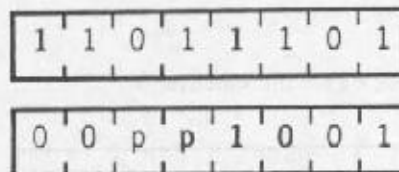
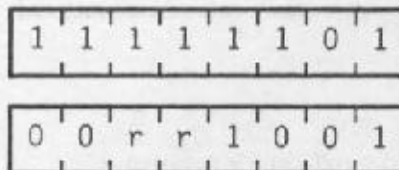
SBC HL, ss

HL ← HL – ss – CY

O conteúdo do par de registradores ss e o flag CARRY são subtraídos do conteúdo do par de registradores HL e o resultado é guardado em HL.

Ciclos: 4
Estados: 15
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado for negativo
Z: ligado se o resultado for zero
H: ligado se não houver BORROW (empréstimo) do bit 12
N: ligado
C: ligado se não houver BORROW (empréstimo)
P/V: ligado se não houver transbordo (overflow)

ADD IX, pp

IX ← IX + pp

O conteúdo do par de registradores pp é somado ao conteúdo do registrador de index (IX) e o resultado é guardado em IX.

Ciclos: 4
Estados: 15
Flags: H, N, C
H: ligado se houver CARRY após o bit 11
N: desligado
C: ligado se houver CARRY do bit 15

ADD IY, rr

IY ← IY + rr

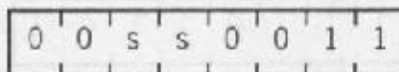
O conteúdo do par de registradores rr é somado ao conteúdo do registrador do index IY, e o resultado é guardado em IY.

Ciclos:	4
Estados:	15
Flags:	H, N, C H: ligado se houver CARRY do bit 11 N: desligado C: ligado se houver CARRY do bit 15

INC ss

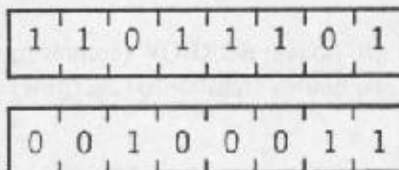
ss ← ss + 1

O conteúdo do par de registradores ss é incrementado.

0 0 s s 0 0 1 1

Ciclos:	1
Estados:	6
Flags:	nenhum

INC IX

IX ← IX + 1

O conteúdo do registrador de index IX é incrementado.

Ciclos:	2
Estados:	10
Flags:	nenhum

INC IY

IY ← IY + 1

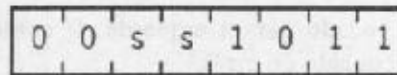
O conteúdo do registrador de index IY é incrementado.

1 1 1 1 1 1 0 1

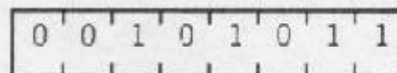
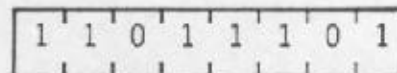
0 0 1 0 0 0 1 1

Ciclos:	2
Estados:	10
Flags:	nenhum

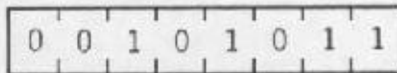
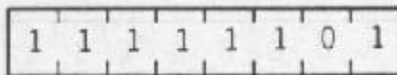
DEC ss

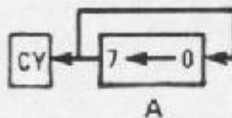
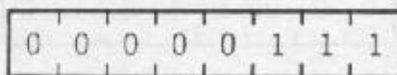
ss ← ss − 1
O conteúdo do par de registradores ss é decrementado.

Ciclos: 1
Estados: 6
Flags: nenhum

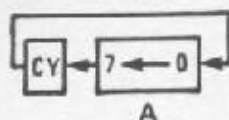
DEC IX

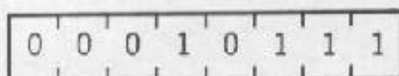
IX ← IX − 1
O conteúdo do registrador de index IX é decrementado.

Ciclos: 2
Estados: 10
Flags: nenhum

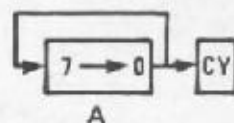
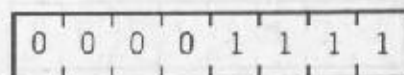
DEC IY

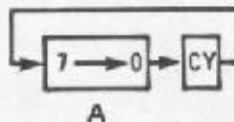
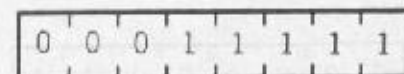
IY ← IY − 1
O conteúdo do registrador de index IY é decrementado.

Ciclos: 2
Estados: 10
Flags: nenhum

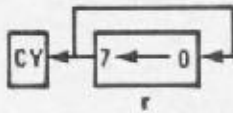
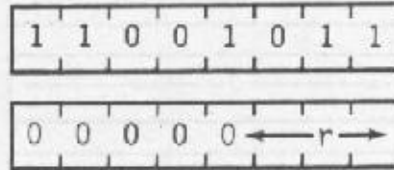
GRUPO DE ROTAÇÃO E DESLOCAMENTO

O conteúdo do Acumulador é rodado para a esquerda. O conteúdo do bit 7 é copiado no flag CARRY e também no bit 0

Ciclos: 1
Estados: 4
Flags: H, N, C
H: desligado
N: desligado
C: dado do bit 7 do Acumulador

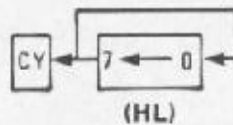
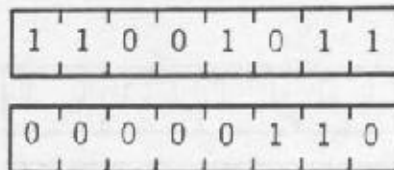
RLA

O conteúdo do Acumulador é rodado para a esquerda. O conteúdo do bit 7 é copiado no flag CARRY e o conteúdo do flag CARRY é copiado do bit 0.

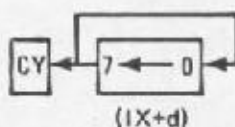
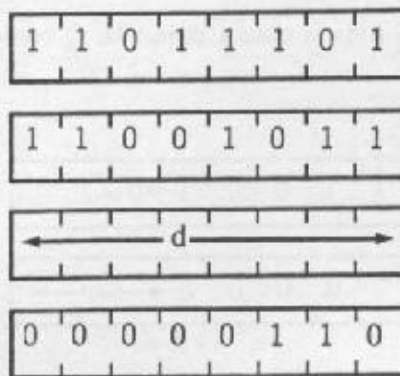
0	0	0	1	0	1	1	1

Ciclos: 1
Estados: 4
Flags: H, N, C
H: desligado
N: desligado
C: dado do bit do Acumulador

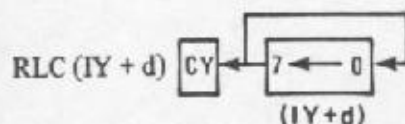
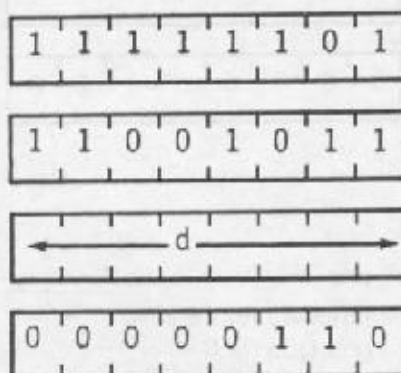
RRCA

O conteúdo do Acumulador é rodado para a direita. O conteúdo do bit 0 é copiado no bit 7 e também no flag CARRY.

Ciclos: 1
Estados: 4
Flags: H, N, C
H: desligado
N: desligado
C: dado do bit 0 do Acumulador

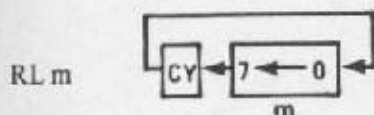
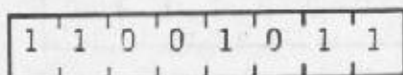
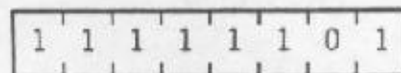
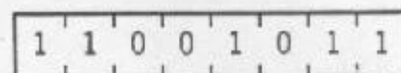
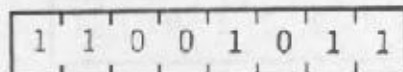
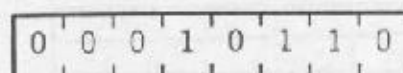
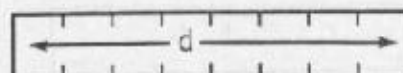
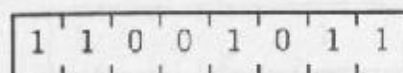
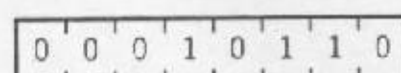
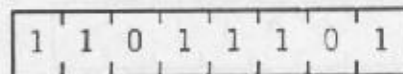
RRA

O conteúdo do Acumulador é rodado para a direita. O conteúdo do bit 0 é copiado no flag CARRY e o conteúdo anterior do flag CARRY é copiado no bit 7.

Ciclos: 1
Estados: 4
Flags: H, N, C
H: desligado
N: desligado
C: dado do bit 0 do Acumulador

RLC r

O conteúdo do registrador r é rodado para a esquerda. O conteúdo do bit 7 é copiado no flag CARRY e também no bit 0.

Ciclos: 2
Estados: 8
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado é negativo
Z: ligado se o resultado é zero
H: desligado
N: desligado
C: dado do bit 7 do registrador fonte
P/V: ligado se paridade par

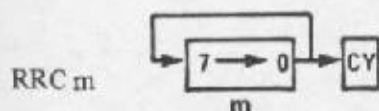
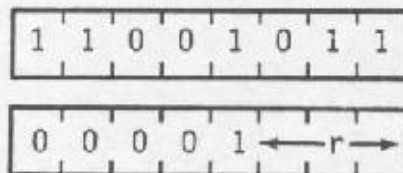
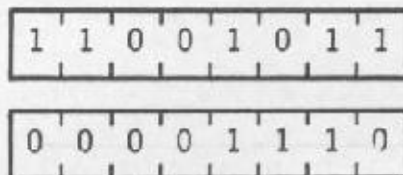
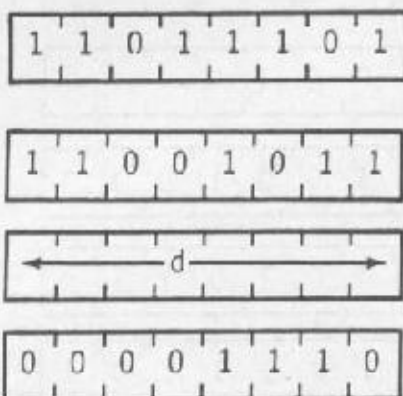
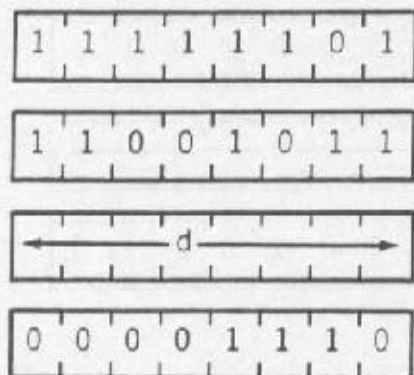
RLC (HL)

O conteúdo do endereço de memória especificado pelo conteúdo do par de registradores HL é rodado para a esquerda. O conteúdo do bit 7 é copiado no flag CARRY e também no bit 0.

Ciclos: 4
Estados: 15
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado for negativo
Z: ligado se o resultado for 0
H: desligado
N: desligado
C: dado do bit 7 do registrador fonte
P/V: ligado se paridade par

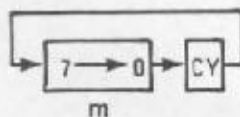
RLC (IX + d)

O conteúdo do endereço de memória, especificado pela soma do conteúdo do registrador de index IX e o deslocamento inteiro em complemento de dois (d), é rodado para a esquerda.

O conteúdo do bit 7 é copiado no flag CARRY e também no bit 0.

Ciclos: 6
Estados: 23
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado for negativo
Z: ligado se o resultado for zero
H: desligado
N: desligado
C: dado do bit 7 do registrador fonte
P/V: ligado se paridade par

O conteúdo do endereço de memória, especificado pela soma do conteúdo do registrador de index IY e o deslocamento inteiro em complemento de dois (d), é rodado para esquerda. O conteúdo do bit 7 é copiado no flag CARRY e também no bit 0.

Ciclos: 6
Estados: 23
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado for negativo
Z: ligado se o resultado for zero
H: desligado
N: desligado
C: dado do bit 7 do registrador fonte
P/V: ligado se paridade par

RL m

O conteúdo do operando m é rodado para a esquerda. O conteúdo do bit 7 é copiado no flag CARRY e o conteúdo anterior do flag CARRY é copiado no bit 0.

Instrução	*Ciclos*	*Estados*
RL r	2	8
RL (HL)	4	15
RL (IX+d)	6	23
RL (IY+d)	6	23

Flags: S, Z, H, N, C, P/V
S: ligado se o resultado for negativo
Z: ligado se o resultado for 0
H: desligado
N: desligado
C: dado do bit 7 do registrador fonte
P/V: ligado se paridade par

RRC m

O conteúdo do operando m é rodado para a direita. O conteúdo do bit 0 é copiado no flag CARRY e também no bit 7.

Instrução	*Ciclos*	*Estados*
RRC r	2	8
RRC (HL)	4	15
RRC (IX+d)	6	23
RRC (IY+d)	6	23

Flags: S, Z, H, N, C, P/V
S: ligado se o resultado for negativo
Z: ligado se o resultado for 0
H: desligado
N: desligado
C: dado do bit 7 do registrador fonte
P/V: ligado se paridade par

O conteúdo do operando m é rodado para a direita. O conteúdo do bit 0 é copiado no flag CARRY e o valor anterior do flag CARRY é copiado no bit 7.

RR r

1	1	0	0	1	0	1	1
0	0	0	1	1	←	r	→

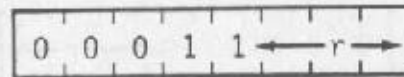
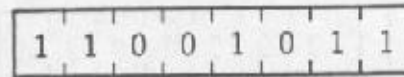
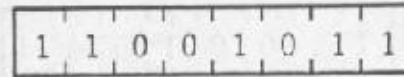
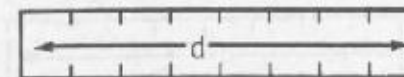
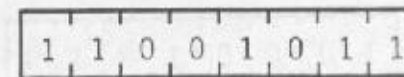
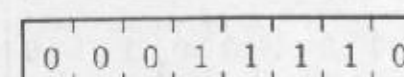
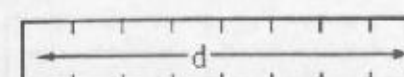
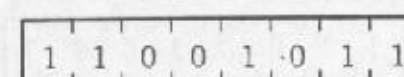
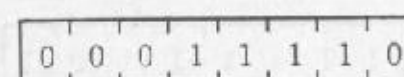
RR (HL)

1	1	0	0	1	0	1	1
0	0	0	1	1	1	1	0

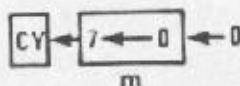
RR (IX+d)

1	1	0	1	1	1	0	1
1	1	0	0	1	0	1	1
←			d				→
0	0	0	1	1	1	1	0

RR (IY+d)

0	0	0	1	1	1	1	0
1	1	0	0	1	0	1	1
←			d				→
0	0	0	1	1	1	1	0

Instrução	*Ciclos*	*Estados*
RR r	2	8
RR (HL)	4	15
RR (IX+d)	6	23
RR (IY+d)	6	23

Flags: S, Z, H, N, C, P/V
S: ligado se o resultado for negativo
Z: ligado se o resultado for 0
H: desligado
N: desligado
C: dado o bit 0 do registrador fonte
P/V: ligado se paridade par

SLA_m

Um deslocamento aritmético para a esquerda é executado no conteúdo do operando m. Bit 0 é desligado. O conteúdo do bit 7 é copiado no flag de CARRY.

SLA r

1	1	0	0	1	0	1	1

0	0	1	0	0	←r→		

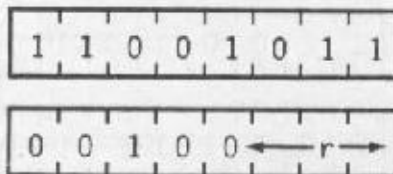
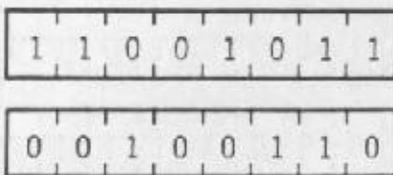
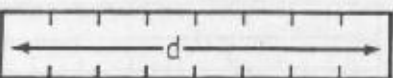
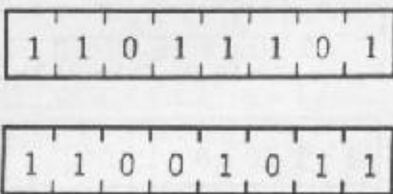
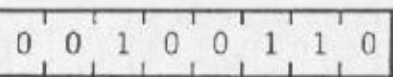
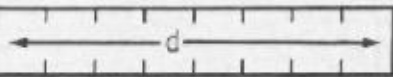
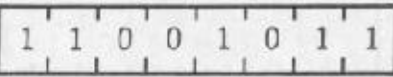
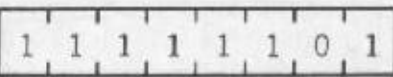
SLA (HL)

1	1	0	0	1	0	1	1
0	0	1	0	0	1	1	0

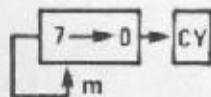
SLA (IX+d)

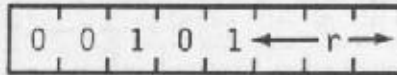
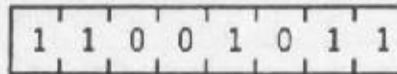
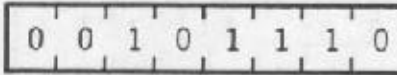
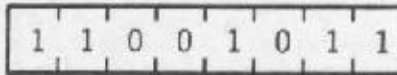
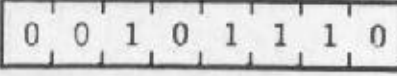
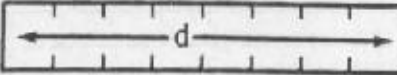
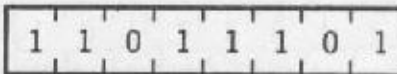
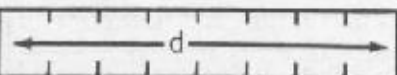
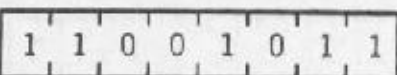
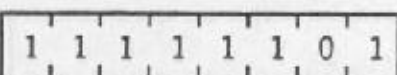
1	1	0	1	1	1	0	1
1	1	0	0	1	0	1	1
←d→							
0	0	1	0	0	1	1	0

SLA (IY+d)

1	1	1	1	1	1	0	1
1	1	0	0	1	0	1	1
←d→							
0	0	1	0	0	1	1	0

Instrução	*Ciclos*	*Estados*
SLA r	2	8
SLA (HL)	4	15
SLA (IX+d)	6	23
SLA (IY+d)	6	23

Flags: S, Z, H, N, C, P/V
S: ligado se o resultado for negativo
Z: ligado se o resultado for 0
H: desligado
N: desligado
C: dado do bit 7
P/V: ligado se paridade par

SRA m

Um deslocamento aritmético para a direita é executado no conteúdo do operando m. O conteúdo do bit 0 é copiado no flag CARRY e o valor anterior do bit 7 não sofre alteração.

SRA r

1 1 0 0 1 0 1 1
0 0 1 0 1 ←r→

SRA (HL)

1 1 0 0 1 0 1 1
0 0 1 0 1 1 1 0

SRA (IX+d)

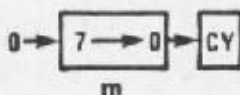
1 1 0 1 1 1 0 1
1 1 0 0 1 0 1 1
←d→
0 0 1 0 1 1 1 0

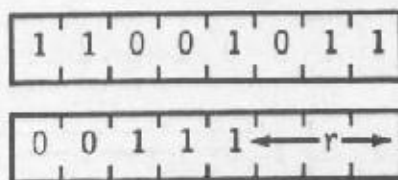
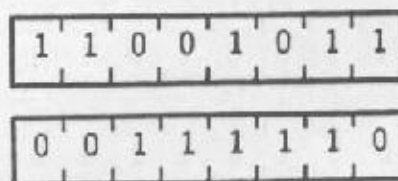
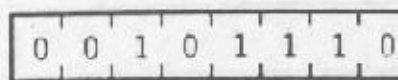
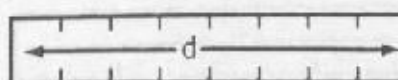
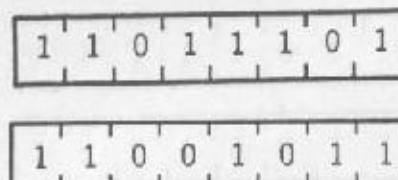
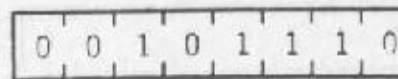
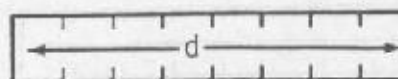
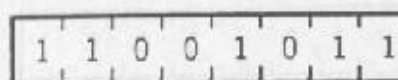
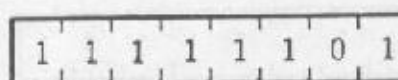
SRA (IY+d)

1 1 1 1 1 1 0 1
1 1 0 0 1 0 1 1
←d→
0 0 1 0 1 1 1 0

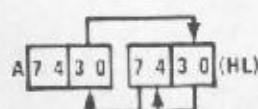
Instrução	*Ciclos*	*Estados*
SRA r	2	8
SRA (HL)	4	15
SRA (IX+d)	6	23
SRA (IY+d)	6	23

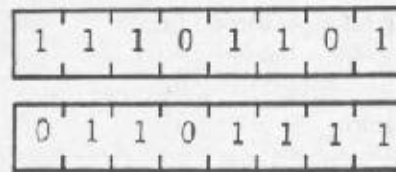
Flags: S, Z, H, N, C, P/V
S: ligado se o resultado for negativo
Z: ligado se o resultado for 0
H: desligado
N: desligado
C: dado do bit 0 do registrador fonte
P/V: ligado se paridade par

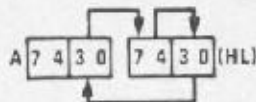
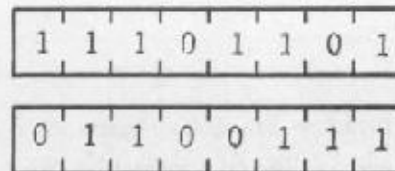
SRL m

O conteúdo do operando m é deslocado para a direita. O conteúdo do bit 0 é copiado no flag CARRY e o bit 7 é desligado.

SRL r	1 1 0 0 1 0 1 1
	0 0 1 1 1 ←r→
SRL (HL)	1 1 0 0 1 0 1 1
	0 0 1 1 1 1 1 0
SRL (IX+d)	1 1 0 1 1 1 0 1
	1 1 0 0 1 0 1 1
	←d→
	0 0 1 0 1 1 1 0
SRA (IY+d)	1 1 1 1 1 1 0 1
	1 1 0 0 1 0 1 1
	←d→
	0 0 1 0 1 1 1 0

Instrução	*Ciclos*	*Estados*
SRA r	2	8
SRA (HL)	4	15
SRA (IX+d)	6	23
SRA (IY+d)	6	23

Flags: S, Z, H, N, C, P/V
S: ligado se o resultado for negativo
Z: ligado se o resultado for 0
H: desligado
N: desligado
C: dado do bit 7 do registrador fonte
P/V: ligado se paridade par

RLD

O conteúdo dos 4 bits mais baixos da localização de memória (HL) é copiado nos 4 bits mais altos da mesma localização de memória. O conteúdo anterior dos 4 bits mais altos é copiado nos 4 bits mais baixos

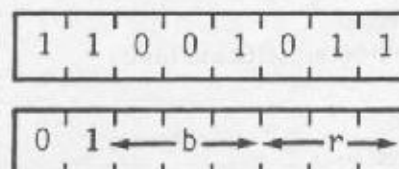
do Acumulador, e o conteúdo anterior dos 4 bits mais baixos do Acumulador é copiado nos 4 bits mais baixos da localização de memória (HL). O conteúdo dos 4 bits mais altos do Acumulador não são afetados.

Ciclos: 5
Estados: 18
Flags: S, Z, H, N, P/V
S: ligado se o Acumulador for negativo após a operação
Z: ligado se o Acumulador for zero após a operação
H: desligado
N: desligado
P/V: ligado se a paridade do Acumulador for par após a operação

RRD

O conteúdo dos 4 bits mais baixos da localização de memória (HL) é copiado nos 4 bits mais baixos do Acumulador. O conteúdo anterior dos 4 bits mais baixos do Acumulador são copiados nos 4 bits mais altos da localização de memória dado por HL, e o conteúdo anterior dos 4 bits mais altos de HL é copiado nos 4 bits mais baixos da localização dada por HL. O conteúdo dos 4 bits mais altos do Acumulador não são afetados.

Ciclos: 5
Estados: 18
Flags: S, Z, H, N, P/V
S: ligado se o Acumulador for negativo após a operação
Z: ligado se o Acumulador for 0 após a operação
H: desligado
N: desligado
P/V: ligado se a paridade do Acumulador for par após a operação

GRUPO DE: LIGAR, DESLIGAR E TESTAR BIT

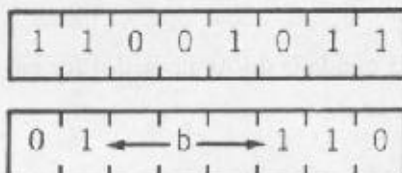
BIT b, r

$Z \leftarrow \overline{r}_b$

Depois da execução desta instrução o flag Z do registrador F conterá o complemento do bit indicado no registrador.

Ciclos: 2
Estados: 8
Flags: S, Z, H, N, P/V
S: desconhecido
Z: ligado se o bit especificado for 0
H: ligado
N: desligado
P/V: desconhecido

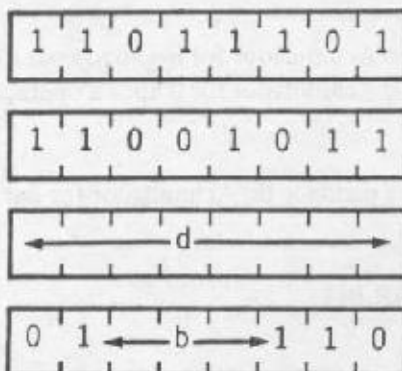
BIT b (HL)

$Z \leftarrow \overline{(HL)}_b$

Depois da execução desta instrução o flag Z no registrador F conterá o complemento do bit indicado na posição dada pelo par de registradores HL.

Ciclos: 3
Estados: 12
Flags: S, Z, H, N, P/V
S: desconhecido
Z: ligado se o bit especificado for 0
H: ligado
N: desligado
P/V: desconhecido

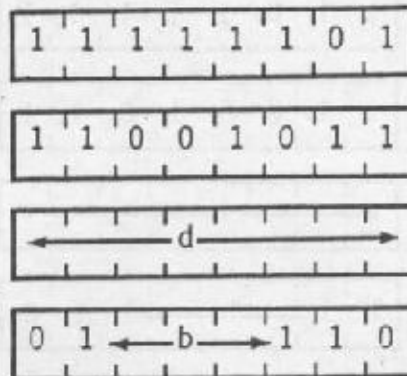
BIT b, (IX + d)

$Z \leftarrow \overline{(IX + d)}_b$

Depois da execução desta instrução, o flag Z no registrador F conterá o complemento do bit indicado dentro do conteúdo da localizado de memória dada pela soma do conteúdo do par de registradores IX e do deslocamento em complemento de dois.

Ciclos: 5
Estados: 20
Flags: S, Z, H, N, P/V
S: desconhecido
Z: ligado se o bit especificado for 0
H: ligado
N: desligado
P/V: desconhecido

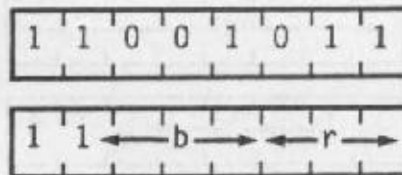
BIT b, (IY + d)

$Z \leftarrow \overline{(IY + d)_b}$

Depois da execução desta instrução, o flag Z do registrador F conterá o complemento do bit indicado dentro do conteúdo da localização de memória dada pela soma do conteúdo do par de registradores IY com o deslocamento em complemento de dois.

Ciclos: 5
Estados: 20
Flags: S, Z, H, N, P/V
S: desconhecido
Z: ligado se o bit especificado for 0
H: ligado
N: desligado
P/V: desconhecido

SET b, r

$r_b \leftarrow 1$

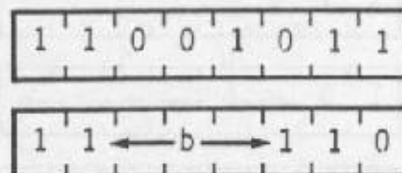
Bit b (qualquer bit de 0 a 7) no registrador r é ligado.

Ciclos: 2
Estados: 8
Flags: nenhum

SET b, (HL)

$(HL)_b \leftarrow 1$

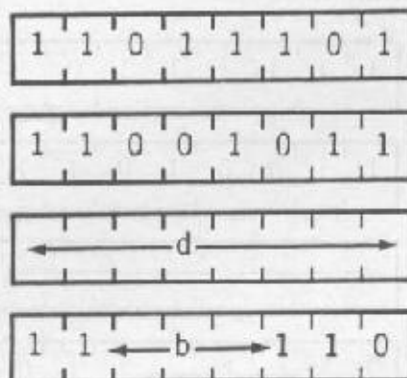
Bit b na localização de memória endereçada pelo conteúdo do par de registradores HL é ligado.

1 1 0 0 1 0 1 1

1 1 ← b → 1 1 0

Ciclos: 4
Estados: 15
Flags: nenhum

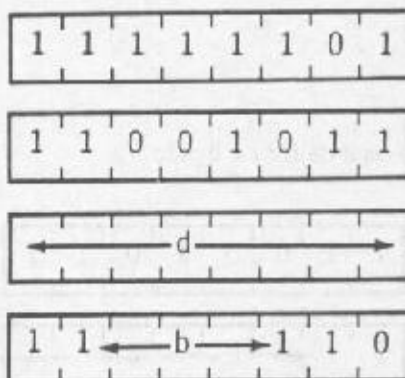
SET b, (IX + d)

$(IX + d)_b \leftarrow 1$

Bit b, na localização de memória endereçada pela soma do par de registradores IX com o deslocamento em complemento de dois, é ligado.

Ciclos: 6
Estados: 23
Flags: nenhum

SET b (IY + d)

$(IY + d)_b \leftarrow 1$

Bit b, na localização de memória endereçada pela soma do par de registradores IY com o deslocamento em complemento de dois, é ligado.

Ciclos: 6
Estados: 23
Flags: nenhum

RES b, m

$S_b \leftarrow 0$

Bit b no operando m é desligado

RES b, r

1 1 0 0 1 0 1 1

1 0 ←b→ ←r→

RES b, (HL)

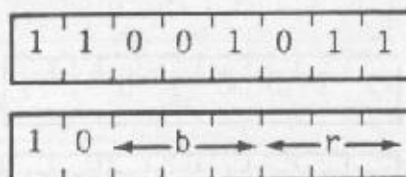
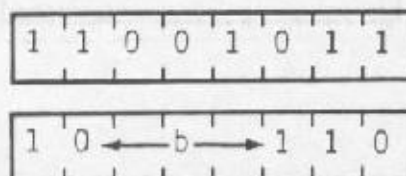
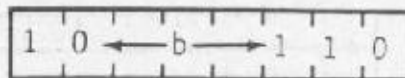
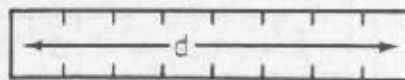
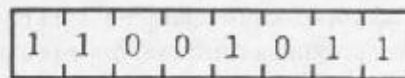
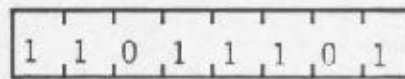
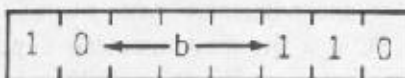
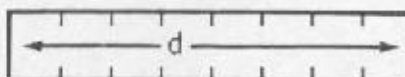
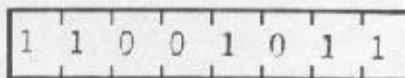
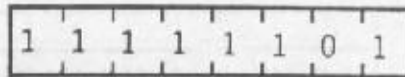
1 1 0 0 1 0 1 1

1 0 ←b→ 1 1 0

RES b, (IX+d)

1	1	0	1	1	1	0	1
1	1	0	0	1	0	1	1
←			d				→
1	0	←	b	→	1	1	0

RES b, (IY+d)

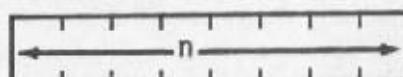
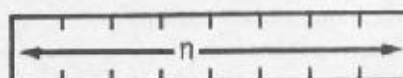
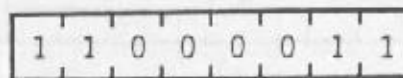
1	1	1	1	1	1	0	1
1	1	0	0	1	0	1	1
←			d				→
1	0	←	b	→	1	1	0

Instrução	*Ciclos*	*Estados*
RES b, r	4	8
RES b, (HL)	4	15
RES b, (IX+d)	6	23
RES b, (IY+d)	6	23

Flags: nenhum

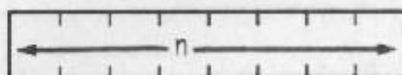
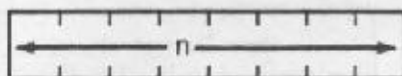
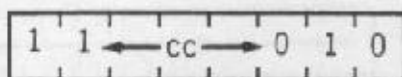
GRUPO DE INSTRUÇÕES DE PULO (JUMP)

JP nn

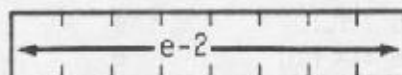
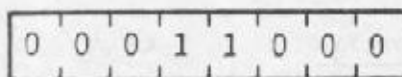
PC ← nn

O operando nn é carregado no par de registradores PC (CONTADOR DO PROGRAMA) e aponta para o endereço da próxima instrução do programa a ser executada.

Ciclos: 3
Estados: 10
Flags: nenhum

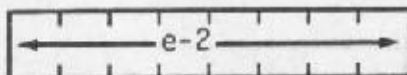
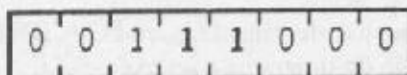
JP cc, nn

Se cc for verdadeiro, PC ← nn
Se a condição cc for verdadeira, a instrução carrega o operando nn no par de registradores PC, e o programa continua com a instrução começando no endereço nn. Se a condição cc for falsa o PC é incrementado normalmente e o programa continua com a próxima instrução.

Ciclos: 3
Estados: 10
Flags: nenhum

JR e

PC ← PC + e
Esta instrução executa pulos incondicionais para outros segmentos de um programa. O valor do deslocamento é somado ao PC e a próxima instrução é pega da localização dada pelo conteúdo do PC. Este pulo é medido do endereço onde está o código de operação da instrução e tem uma área de atuação de −126 a +129 bytes.

Ciclos: 3
Estados: 12
Flags: nenhum

JR C, e

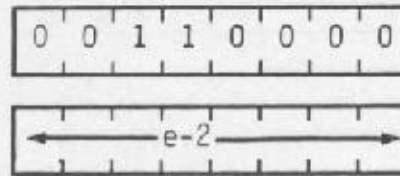
Se C = 0, continue
Se C = 1, PC ← PC + e
Esta instrução executa pulos condicionais para outros segmentos de um programa dependendo do resultado do teste do flag CARRY. Se o flag estiver ligado, o valor do deslocamento é somado ao PC, e a próxima instrução é pega desta localização dada pelo novo conteúdo do PC. Se o flag estiver desligado, a próxima instrução é tomada a partir da localização seguinte a esta instrução.

Se a condição for satisfeita:

Ciclos: 3
Estados: 12

Se a condição não for satisfeita:

Ciclos: 2
Estados: 7
Flags: nenhum

JR NC, e

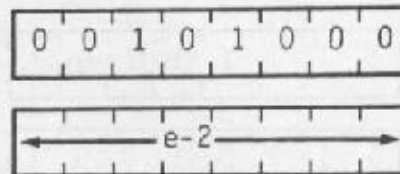
Se C = 1, continue
Se C = 0, PC ← PC + e
Esta instrução executa um pulo condicional para outros segmentos de um programa dependendo do resultado do teste do flag CARRY. Se o flag está desligado, o valor do deslocamento *e* é somado ao PC, e a próxima instrução é pega da localização especificada pelo novo conteúdo do PC. Se o flag estiver ligado, a próxima instrução a ser executada é tomada da posição logo a seguir da instrução.

Se a condição for satisfeita

Ciclos: 3
Estados: 12

Se a condição não for satisfeita

Ciclos: 2
Estados: 7
Flags: nenhum

JR Z, e

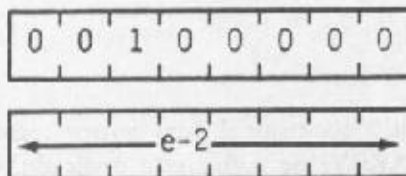
Se Z = 0, continue
SE Z = 1, PC ← PC + e
Se o flag de zero estiver ligado, o valor do deslocamento é somado ao PC e a próxima instrução é pega na localização designada pelo novo conteúdo do PC. Se o flag de zero estiver desligado, a próxima instrução a ser executada é pega da localização a seguir desta instrução.

Se a condição for satisfeita

Ciclos: 3
Estados: 12

Se a condição não for satisfeita

Ciclos: 2
Estados: 7
Flags: nenhum

JR NZ, e

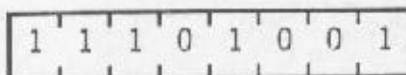
Se Z = 1, continue
Se Z = 0, PC ← PC + e
Se o flag de zero estiver desligado, o valor do deslocamento é somado ao PC, e a próxima instrução é pega na localização designada pelo novo conteúdo do PC. Se o flag de zero estiver ligado, a próxima instrução a ser executada é pega da localização a seguir a esta instrução.

Se a condição for satisfeita

Ciclos: 3
Estados: 12

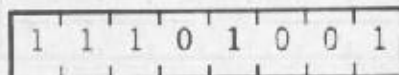
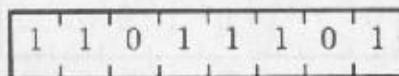
Se a condição não for satisfeita

Ciclos: 2
Estados: 7
Flags: nenhum

JP (HL)

PC ← HL
O PC é carregado com o conteúdo do par de registradores HL. A próxima instrução a ser pega será a da localização designada pelo novo conteúdo do PC.

Ciclos: 1
Estados: 4
Flags: nenhum

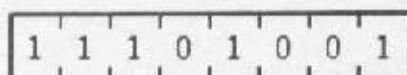
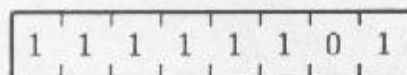
JP (IX)

PC ← IX
O PC é carregado com o conteúdo do par de registradores IX. A próxima instrução é pega da localização designada pelo novo conteúdo do PC.

Ciclos: 2
Estados: 8
Flags: nenhum

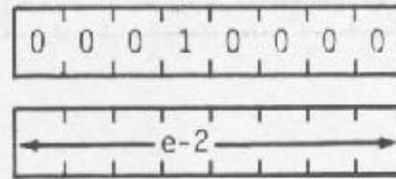
JP (IY)

PC ← IY
O PC é carregado com o conteúdo do par de registradores IY. A próxima instrução é pega da localização designada pelo novo conteúdo do PC.

1 1 1 1 1 1 0 1

1 1 1 0 1 0 0 1

Ciclos: 2
Estados: 8
Flags: nenhum

DJNZ, e

O registro B é decrementado e se um valor diferente de 0 permanecer, o valor do deslocamento *e* é somado ao PC. A próxima instrução é pega da localização designada pelo novo conteúdo do PC. Se o resultado do decremento de B deixá-lo com o valor 0, a próxima instrução a ser executada é pega da localização a seguir a esta instrução.

Se B ≠ 0:

Ciclos: 3
Estados: 13

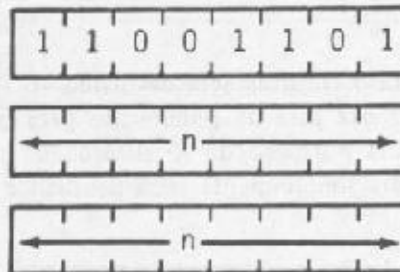
Se B = 0

Ciclos: 2
Estados: 8
Flags: nenhum

GRUPO DE CHAMADA E RETORNO

CALL nn

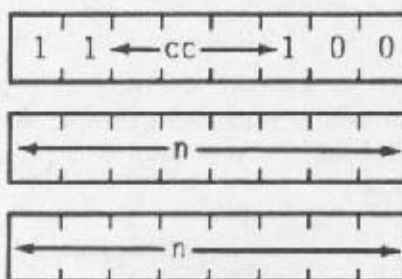
$(SP-1) \leftarrow PC_H, (SP-2) \leftarrow PC_L, PC \leftarrow nn$

Depois de colocar o conteúdo atual do PC no topo da memória de pilha, os operandos nn são carregados no PC, que apontará para o endereço de memória onde o primeiro código de uma sub-rotina será pega. Note que como é uma instrução de 3 bytes, o PC terá sido incrementado de 3 antes de ser salvo na pilha.

Ciclos: 5
Estados: 17
Flags: nenhum

CALL cc, nn

Se cc for verdadeiro:

$(SP-1) \leftarrow PC_H, (SP-2) \leftarrow PC_L, PC \leftarrow nn$

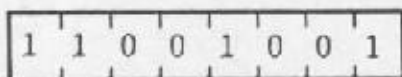
Se a condição cc for verdadeira, esta instrução colocará o conteúdo atual do PC na pilha externa e carregará, então, o operando nn no PC para que este enderece o primeiro código de operação de uma sub-rotina.

Se cc for verdadeira:

Ciclos: 5
Estados: 17

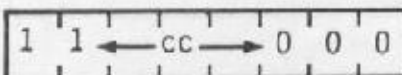
Se cc for falso:

Ciclos: 3
Estados: 10
Flags: nenhum

RET

$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP + 1)$
O controle é devolvido ao fluxo original do programa fazendo com que o conteúdo prévio do PC que está na pilha volte para o PC. No próximo ciclo de máquina o processador irá pegar a próxima instrução da localização de memória apontada pelo PC atual.

Ciclos: 3
Estados: 10
Flags: nenhum

RET cc

Se cc for verdadeiro:
$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP + 1)$
Se a condição cc for verdadeira o controle será devolvido ao fluxo original do programa fazendo com que o conteúdo prévio do PC que está na pilha volte para o PC. No próximo ciclo de máquina o processador irá pegar a próxima instrução da localização de memória apontada pelo PC atual. Se a condição cc for falsa, o PC será simplesmente incrementado e o programa continuará com a próxima instrução.

Se cc for verdadeiro:

Ciclos: 3
Estados: 11

Se cc for falso:

Ciclos: 1
Estados: 5
Flags: nenhum

RETI

Retorno de uma interrupção

Esta instrução é usada no fim de uma rotina de tratamento de uma interrupção para:

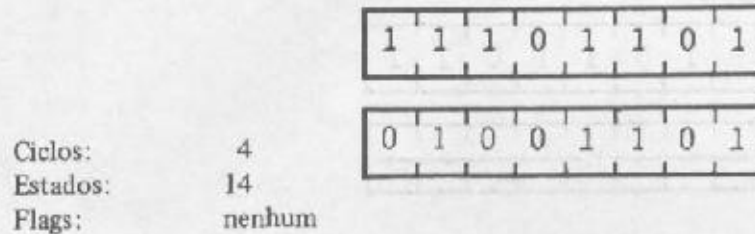
1. Restaurar o conteúdo do PC
2. Sinalizar a um dispositivo de entrada/saída que a rotina de interrupção terminou.

A instrução RETI facilita o encadeamento de interrupções permitindo que dispositivos com maior prioridade suspendam o tratamento de rotinas de menor prioridade.

Esta instrução também desliga os flip-flops IFF1 e IFF2.

1	1	1	0	1	1	0	1

0	1	0	0	1	1	0	1

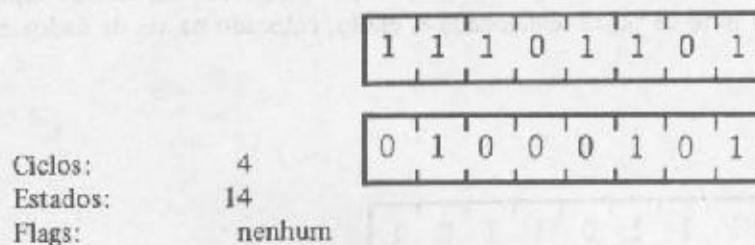
Ciclos: 4
Estados: 14
Flags: nenhum

RETN

Retorno de interrupções não mascaráveis.

Usada ao final de uma sub-rotina de tratamento de uma interrupção não mascarável, a instrução executa um retorno incondicional que funciona da mesma maneira que a instrução RET. O controle é devolvido ao fluxo original do programa; no próximo ciclo de máquina o processador pega a próxima instrução da localização de memória apontada pelo PC. Também o estado de IFF2 é copiado em IFF1 para o estado que existia antes da aceitação de um NMI.

1	1	1	0	1	1	0	1

0	1	0	0	0	1	0	1

Ciclos: 4
Estados: 14
Flags: nenhum

RST p

$(SP-1) \leftarrow PC_H, (SP-2) \leftarrow PC_L, PC_H \leftarrow 0, PC_L \leftarrow P$

O conteúdo atual do PC é salvo na pilha e a localização na página zero da memória é dada pelo operando P que é carregado no PC. A execução do programa começa, então, no novo endereço dado pelo PC. A instrução de RESTART permite um pulo para um dos oito endereços mostrados na tabela a seguir. O operando P é montado no código objeto usando o estado t correspondente.

1	1	←	t	→	1	1	1

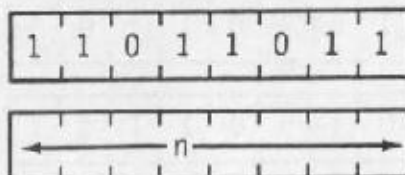
p	t
00H	000
08H	001
10H	010
18H	011
20H	100
28H	101
30H	110
38H	111

Ciclos: 3
Estados: 11
Flags: nenhum

GRUPO DE ENTRADA E SAÍDA

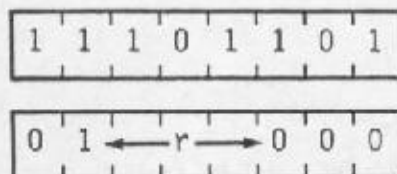
IN A, (n)

A ← (n)

O operando n é colocado na metade inferior da via de endereços para selecionar o dispostivo de entrada/saída (E/S) em um dos 256 endereços possíveis. O conteúdo do Acumulador também aparece na metade superior da via de endereço neste tempo. Um byte da porta selecionada é, então, colocado na via de dados e escrito no Acumulador do processador.

Ciclos: 3
Estados: 11
Flags: nenhum

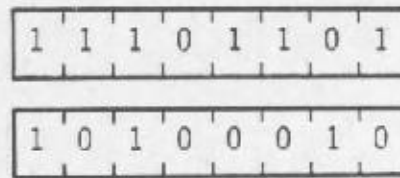
IN r, (C)

r ← (C)

O conteúdo do registrador C é colocado na metade inferior da via de endereços para selecionar uma das 256 portas de entrada/saída possíveis. O conteúdo do registrador B é colocado na metade superior da via de endereços neste tempo. Um byte da porta selecionada é, então, colocado na via de dados e escrito no registrador r do processador.

Ciclos: 3
Estados: 12
Flags: S, Z, H, N, P/V
S: ligado se o dado de entrada for negativo
Z: ligado se o dado de entrada for 0
H: desligado
N: desligado
P/V: ligado se o dado de entrada tiver paridade par

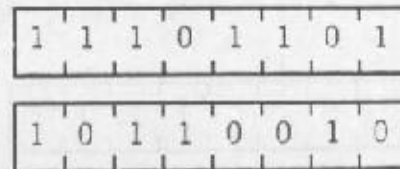
INI

(HL) ← (C), B ← B – 1, HL ← HL + 1

O conteúdo do registrador C é colocado na metade inferior da via de endereços para selecionar uma das 256 portas possíveis. O registrador B pode ser usado como contador e o seu conteúdo é colocado na metade superior da via de endereços. Um byte da porta selecionada é colocado na via de dados e escrito na localização de memória correspondente. Finalmente, o contador é decrementado e o par de registradores HL é incrementado.

Ciclos:	4
Estados:	16
Flags:	S, Z, H, N, P/V
	S: desconhecido
	Z: ligado se B − 1 = 0
	H: desconhecido
	N: ligado
	P/V: desconhecido

INIR

(HL) ← (C), B ← B − 1, HL ← HL + 1

O conteúdo do registrador C é colocado na metade inferior da via de endereços para selecionar uma das 256 portas possíveis. O registrador B é usado como contador e seu conteúdo é colocado na metade superior da via de endereços. Um byte é selecionado, colocado na via de dados e escrito no processador central. O conteúdo do par de registradores HL é colocado no endereço, e o byte de entrada é escrito na localização de memória correspondente. O contador é decrementado e o par de registradores HL é incrementado. Quando B chegar a 0, a instrução é terminada. Se B não é 0, o PC é decrementado por dois e a instrução é repetida. As interrupções são reconhecidas após cada transferência.

Se B ≠ 0

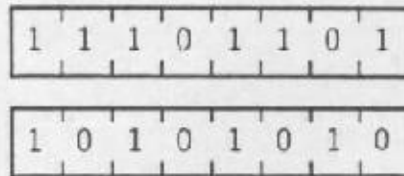
Ciclos:	5
Estados:	21

Se B = 0

Ciclos:	4
Estados:	16
Flags:	S, Z, H, N, P/V
	S: desconhecido
	Z: ligado
	H: desconhecido
	N: ligado
	P/V: desconhecido

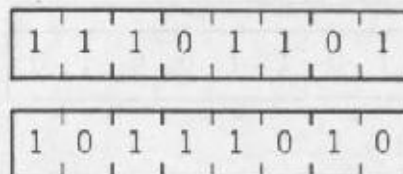
IND

(HL) ← (C), B ← B − 1, HL ← HL − 1

O conteúdo do registrador C é colocado na metade inferior da via de endereço para selecionar um componente de E/S. O registrador B pode ser usado como contador e o seu conteúdo é colocado na metade superior da via de endereços. Um byte da porta selecionada é colocado na via de dados e escrito no processador. O conteúdo do par de registradores HL é colocado na via de endereços e o byte de entrada é, então, escrito na localização de memória correspondente. Finalmente o contador e o par de registradores são decrementados.

Ciclos: 4
Estados: 16
Flags: S, Z, H, N, P/V
S: desconhecido
Z: ligado se B − 1 = 0
H: desconhecido
N: ligado
P/V: desconhecido

INDR

(HL) ← (C), B ← B − 1, HL ← HL − 1

O conteúdo do registrador C é colocado na metade inferior da via de endereços para selecionar o dispositivo de E/S. O registrador B é usado como contador e seu conteúdo é colocado na metade superior da via de endereços. Um byte da porta selecionada é colocado na via de dados e escrito no processador. O conteúdo do par de registradores HL é colocado na via de endereços e o byte de entrada é escrito na localização da memória correspondente. O par de registradores HL e o contador B são decrementados. Quando B chegar a 0, a instrução é terminada. Se B não é 0, o PC é decrementado por 2 e a instrução é repetida. As interrupções serão reconhecidas após cada transferência de dados.

Se B ≠ 0

Ciclos: 5
Estados: 21

Se B = 0

Ciclos: 4
Estados: 16
Flags: S, Z, H, N, P/V
S: desconhecido
Z: ligado
H: desconhecido
N: ligado
P/V: desconhecido

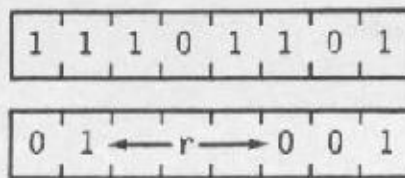
OUT (n), A

(n) ← A

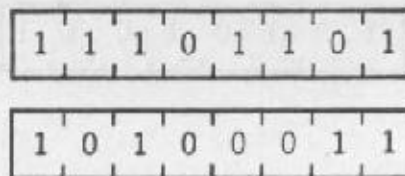
O operando n é colocado na metade inferior da via de endereços para selecionar o dispositivo de E/S. O conteúdo do Acumulador aparece na metade superior da via de endereços. Então o byte contido no Acumulador é colocado na via de dados e escrito no dispositivo selecionado.

Ciclos: 3
Estados: 11
Flags: nenhum

OUT (C), r

(C) ← r
O conteúdo do registrador (C) é colocado na metade inferior da via de endereços para selecionar o dispositivo de E/S. O conteúdo do registrador B é colocado na metade superior da via de endereços. O byte contido no registrador r é colocado na via de dados e escrito no dispositivo de E/S.

Ciclos: 3
Estados: 12
Flags: nenhum

OUTI

(C) ← (HL), B ← B – 1, HL ← HL + 1
O conteúdo do par de registrador HL é colocado na via de endereços para selecionar uma localização de memória. O byte contido nesta localização de memória é temporariamente guardado no processador. Depois de decrementado o registrador B, o conteúdo do registrador C é colocado na metade inferior da via de endereços para selecionar o dispositivo de E/S. O registrador B pode ser usado como contador, e seu valor decrementado é colocado na metade inferior da via de endereços. O byte a ser enviado é colocado na via de dados e escrito no dispositivo selecionado. Finalmente o par de registradores HL é incrementado.

Ciclos: 4
Estados: 16
Flags: S, Z, H, N, P/V
S: desconhecido
Z: ligado se B – 1 = 0
H: desconhecido
N: ligado
P/V: desconhecido

OTIR

(C) ← (HL), B ← B – 1, HL ← HL + 1
O conteúdo do par de registradores HL é colocado na via de endereços para selecionar a localização de memória. O byte contido nesta localização de memória é temporariamente guardado no processador. Depois que o contador é decrementado (registrador B), o conteúdo do registrador C é colocado na metade inferior da via de endereços para selecionar o dispositivo de E/S. O registrador B pode ser usado como contador e o seu valor é colocado na metade superior da via de endereços. O byte a ser enviado é colocado na via de dados e escrito no dispositivo de E/S selecionado; logo, o par de registradores HL é incrementado. Se o registrador B não for 0, o PC será decrementado por dois e a instrução será repetida. Se B for 0, a instrução estará terminada. As interrupções serão reconhecidas após cada transferência de dados.

1	1	1	0	1	1	0	1

1	0	1	1	0	0	1	1

Se B ≠ 0

Ciclos: 5
Estados: 21

Se B = 0

Ciclos: 4
Estados: 16
Flags: S, Z, H, N, P/V
S: desconhecido
Z: ligado
H: desconhecido
N: ligado
P/V: desconhecido

OUTD

(C) ← (HL), B ← B – 1, HL ← HL – 1
O conteúdo do par de registradores HL é colocado na via de endereços para selecionar uma localização de memória. O byte contido nesta localização de memória é temporariamente guardado no processador. Depois que o contador é decrementado, o conteúdo do registrador C é colocado na metade inferior da via de endereço para selecionar o dispositivo de E/S. O byte a ser enviado é colocado na via de dados e escrito no dispositivo E/S selecionado. Finalmente o par de registradores HL é decrementado.

1	1	1	0	1	1	0	1

1	0	1	0	1	0	1	1

Ciclos:
Estados: S, Z, H, N, P/V
S: desconhecido
Z: ligado se B – 1 = 0
H: desconhecido
N: ligado
P/V: desconhecido

OUTDR

(C) ← (HL), B ← B – 1, HL ← HL – 1
O conteúdo do par de registradores HL é colocado na via de endereços para selecionar uma localização de memória. O byte contido nesta localização de memória é temporariamente guardado no processador. Depois do contador ser decrementado, o conteúdo do registrador C é colocado na metade inferior da via de endereços para selecionar o dispositivo de E/S. O registrador B pode ser usado como contador e, depois de decrementado seu valor, é colocado na metade superior da via de endereços. O byte a ser enviado é, então, colocado na via de dados e escrito no dispositivo selecionado. O par de registradores HL é, então, decrementado. Se o registrador B não for 0, o PC será decrementado por 2, e a instrução será repetida. Se o registrador B é 0, então a instrução será terminada. As interrupções serão reconhecidas depois de cada transferência de dados.

1	1	1	0	1	1	0	1

1	0	1	1	1	0	1	1

Se $B \neq 0$

Ciclos: 5
Estados: 21

Se $B = 0$

Ciclos: 4
Estados: 16
Flags: S, Z, N, H, P/V
S: desconhecido
Z: ligado
H: desconhecido
N: ligado
P/V: desconhecido

CAPÍTULO 4

CONSTRUA O SEU PRÓPRIO COMPUTADOR – COMECE COM O BÁSICO

O computador a ser construído por meio do projeto já descrito será chamado PAZ (Processador de Aplicações Z80). Construir um computador desde o início é tanto educativo como útil. Eu explicarei detalhadamente cada seção do processo de construção. Cada passo deveria ser testado antes de prosseguir para o próximo estágio, porém isto não é possível em todos os casos, contudo existe um efeito benéfico ao se tomar esta direção. Geralmente bons projetos não funcionam porque o nível de construção está além da habilidade do montador.

Como a maioria dos montadores não possuem equipamento de teste sofisticado, tais como osciloscópio, analisadores lógicos, procurarei manter as rotinas de teste as mais simples possíveis, dividindo o PAZ em áreas lógicas para análise e teste (e usando componentes testados). Os problemas podem ser detectados em estágios iniciais e consertados mais facilmente.

A construção inicial do PAZ se constituirá de uma configuração operacional mínima. É importante que ela funcione antes de anexarmos algum circuito adicional. Todo esforço será feito para que o leitor se familiarize com os componentes de cada seção e a filosofia do projeto. Enquanto for necessário que se monte todos os componentes desta configuração mínima, para que se possa testar o funcionamento do processador central, um pré-teste de subconjuntos deverá minimizar os erros de fiação.

O PAZ básico divide-se em quatro partes principais: as barras de dados, endereço e controle, decodificação de memória e entrada/saída, e registradores de entrada/saída. Elas serão posteriormente, divididas a nível de componentes. Os esquemas incluem uma explicação completa das suas funções lógicas e os procedimentos de teste são apresentados após cada construção.

O PROCESSADOR

A figura 4.1 mostra o diagrama em blocos detalhados do computador PAZ.

I. Lógica das Barras de Dados, Endereço e Controle

A. Geração do Clock (Relógio)

O computador PAZ roda em 2,5 MHz. Diferentemente do 8080A, o Z80 necessita de um clock de uma só fase e pode funcionar desde CC (corrente contínua) até 2,5 MHz (o Z80A funciona com 4MHz). A figura 4.2 mostra os tempos dos ciclos básicos do computador.

Cada operação básica do computador (Mn) acontece em três ou seis períodos de clock. A figura 4.2 mostra um ciclo de instrução típico que consiste de três ciclos de máquina: busca (fetch), leitura da memória e escrita da memória. Depois da busca do código de operação da instrução durante o ciclo M1, os ciclos subseqüentes movem o dado entre a memória e o processador central.

As figuras 4.3a e 4.3b mostram o projeto de dois circuitos de clock possíveis para o Z80. Ambos os circuitos têm um resistor de pull-up para +5V. Isto irá satisfazer os requisitos CC e CA do Z80, mas é melhor usar uma porta inversora separada para executar o pull-up, qualquer que seja a técnica de oscilação usada.

O circuito oscilador controlado a cristal da figura 4.3a é preferível se o tempo de execução tem de ser constante. Assim o circuito da figura 4.3b deverá ser evitado se o computador for usado como contador de eventos, porém é de grande ajuda inicialmente, pois permite diminuir a freqüência de operação do processador central. Se for necessário fazer o clock funcionar passo a passo, o circuito da figura 4.4 poderá ser usado. Para que ocorra um número de ciclos de clock necessários à execução de uma única instrução, necessitaria de um incontável número de apertos de botão para acompanhar a execução de um programa.

Um método mais fácil de diagnóstico seria usar um passo a passo para a instrução. O circuito mostrado na figura 4.5 não faz parte do esquemático final do PAZ, porque o seu uso só se faz necessário se realmente o montador tiver um problema e precisar seguir o fluxo do programa instrução por instrução. Esta função de passo a passo de instrução é conseguida usando-se os sinais de controle gerados pelo Z80 durante a execução do programa. Os dois sinais que interessam são o $\overline{\text{M1}}$ e o $\overline{\text{WAIT}}$. O sinal $\overline{\text{M1}}$ é de saída e o $\overline{\text{WAIT}}$ de entrada. Como mostrado pela figura 4.6, $\overline{\text{M1}}$ vai ao nível lógico zero no início de cada ciclo de busca de instrução. O sinal $\overline{\text{M1}}$ significa que o microprocessador acabou de completar uma instrução e está começando uma outra. O objetivo é fazer com que o microprocessador pare antes de executar a próxima instrução.

A entrada de $\overline{\text{WAIT}}$ do Z80 faz justamente isto. Um zero lógico aplicado nesta entrada suspende a execução do programa e faz com que o computador pare indefinidamente no ciclo $\overline{\text{M1}}$. Durante T_2 o processador central amostra a linha de $\overline{\text{WAIT}}$ na subida do clock. Se nesta hora a entrada de $\overline{\text{WAIT}}$ estiver em zero, um ciclo adicional de $\overline{\text{WAIT}}$ será inserido, e a linha será amostrada outra vez. O processador central permanecerá neste modo até que a linha $\overline{\text{WAIT}}$ vá para o nível lógico 1. Note que isto não é a mesma coisa que um comando de HALT.

O verdadeiro objetivo destes sinais é permitir que periféricos ou memórias mais lentos possam ser usados com processadores muito rápidos. Estados extras de $\overline{\text{WAIT}}$ podem ser inseridos quando necessários para que elementos mais lentos possam ser acessados pelo processador. O circuito da figura 4.5 nos permite controlar o estado de $\overline{\text{WAIT}}$ e executar apenas uma instrução com cada aperto do botão. A saída de CI 1, pino 8 ($\overline{\text{WAIT}}$), está normalmente baixa, ocasionando com isto uma espera indefinida. Quando o botão é apertado, um único pulso ativa IC 2, que é um flip-flop do tipo D. A duração deste pulso é irrelevante, porque o flip-flop só é acionado na subida do clock. Ao ser pressionado o botão, CI 2 é ligado fazendo com que a linha de $\overline{\text{WAIT}}$ vá para um, isto faz com que o processador comece a executar uma instrução. Mas quando este começar a executar a próxima instrução, ou seja, o próximo ciclo de busca, $\overline{\text{M1}}$ vai a zero como antes, o que ativa o mono-estável. Quando CI 3 è ativado, desativa CI 2 e o processador central volta à condição de espera (WAIT), até que o botão volte a ser pressionado outra vez.

Este processo de passo a passo não tem grande valia a menos que possamos monitorar o conteúdo de todos os registradores e determinar o que o processador está tentando fazer. Para que se possa fazer isto, o PAZ deverá estar completamente operacional e estar rodando um programa monitor de parada o qual permite ao usuário fazer o passo a passo com uma rotina de software. Nós discutiremos este programa posteriormente.

Este fato não tem muito sentido para uma pessoa que tem um computador funcionando parcialmente. Embora fosse bom vermos o conteúdo dos registradores, é impossível fazê-lo sem que o processador possa rodar uma rotina de dump e de mostrar na tela. Isto não pode ser feito com o circuito da figura 4.5. Porém é possível olhar o conteúdo das barras de dados e endereço enquanto o processador estiver parado. Isto já deverá dar uma boa indicação se o computador está funcionando corretamente.

Muitos instrumentos podem ser usados para lermos os níveis TTL nas barras. Um osciloscópio ou um voltímetro com alta impedância podem ser usados, porém um mostrador das barras é uma idéia melhor. Estes circuitos estão incluídos como ajuda e não são necessários para a operação do PAZ.

Basicamente o circuito da figura 4.7a é um circuito simples com LED que é duplicado 16 vezes para a barra de endereços e 8 vezes para a de dados. Como o Z80 tem a capacidade de suprir corrente para apenas uma carga TTL temos de colocar o circuito dos mostradores após os circuitos de buffer das barras.

Algumas vezes existe a necessidade de se monitorar um ponto no circuito e ver-se as mudanças de estado. O circuito da figura 4.7a consegue apenas detectar as variações lentas enquanto os pulsos rápidos como M1 não seriam vistos.

Para se monitorar a ocorrência destes pulsos rápidos, especialmente se não dispomos de um osciloscópio, é aconselhável montar o circuito da figura 4.7b. Esta ponta de prova lógica é adequada para a maioria das aplicações, mas deve se tomar cuidado quando do seu uso, pois ela não detecta circuito aberto e o detector de pulso só dispara na descida de qualquer transição. Se isto representar algum problema, acrescente o circuito opcional 7486, que permitirá a detecção tanto na subida como na descida do pulso.

A ponta de prova lógica ou detector de nível similar (osciloscópio, voltímetro digital etc.) são necessários para que os circuitos sejam testados isoladamente.

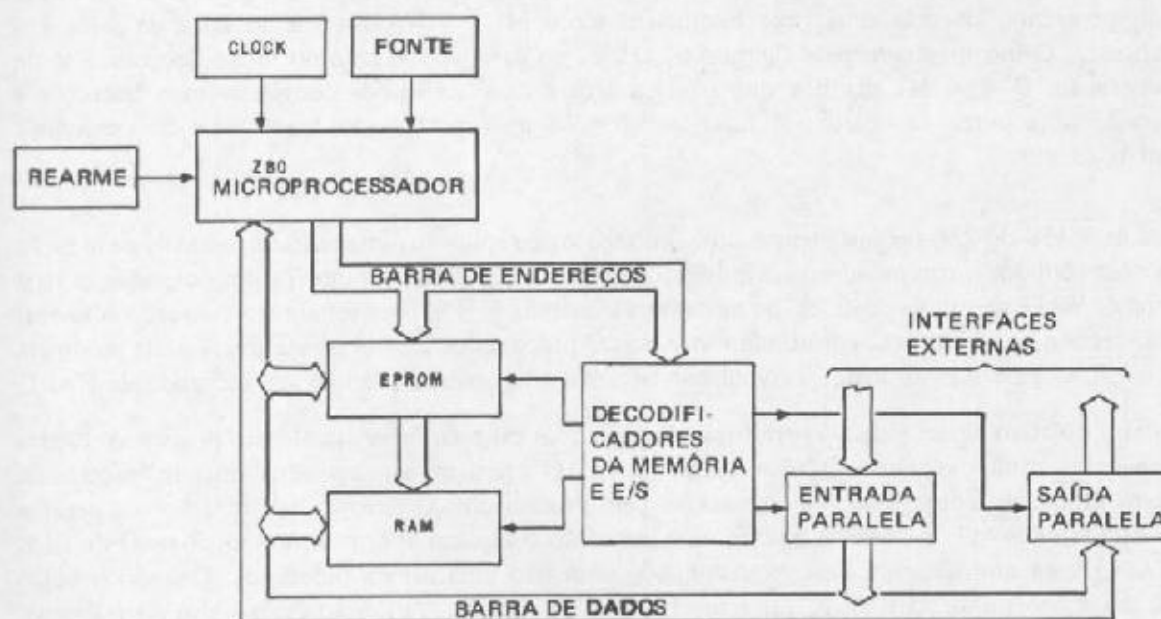



Figura 4.1 Diagrama em bloco do sistema PAZ Mínimo.

Figura 4.2 Um exemplo do diagrama de tempos de um ciclo de instrução

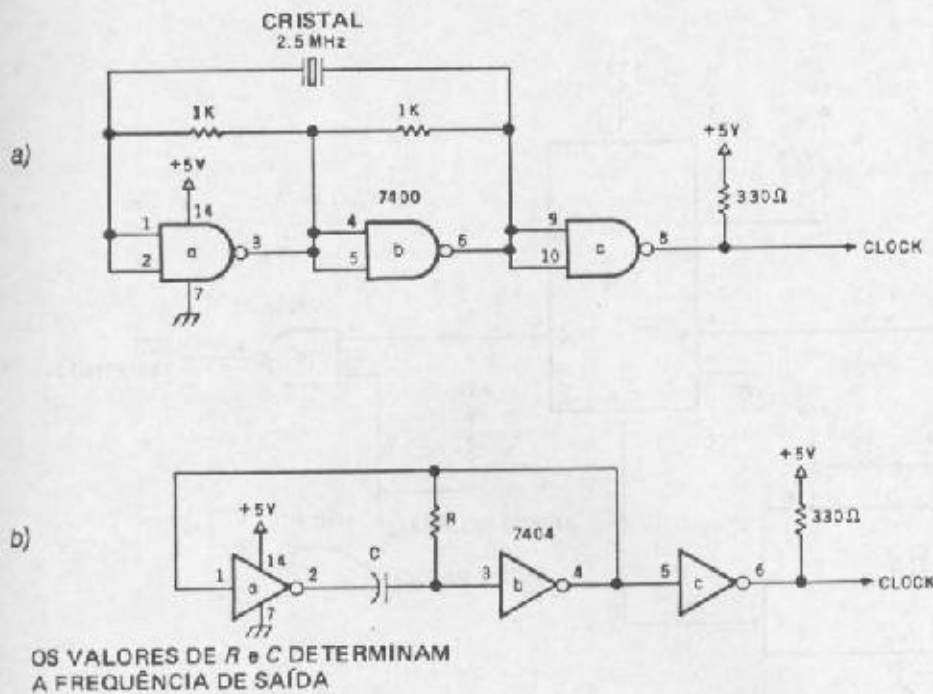



Figura 4.3 Circuitos típicos de clock de 2,5 MHz para o Z80.
a) Controlado a cristal.
b) Freqüência variável.

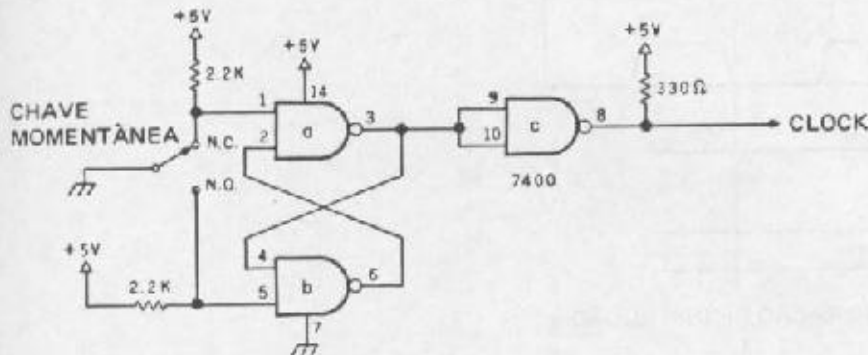



Figura 4.4 Circuito gerador de um só clock.

B. Circuito de Rearme (Reset)

Geralmente ignorada, a função de rearme é um dos controles que mais se precisa num computador. Sua importância é imediatamente reconhecida quando se está rodando um programa incorretamente. O comando de rearme no Z80 pára a execução e carrega o contador de programa com 00 hexadecimal (mais baixo endereço de memória). Isto permite ao programador reiniciar o programa. Quando combinado com o circuito de passo a passo de instrução, os programas podem ser começados, parados e começados outra vez a qualquer tempo.

O sinal de rearme pode ser manual, automático ou uma combinação de ambos. A figura 4.8a é um circuito padrão de rearme. Sua saída está normalmente alta enquanto o botão estiver sendo pressionado e só começará a execução após sua liberação. O rearme manual é uma necessidade para teste inicial do programa e este circuito é empregado no PAZ.

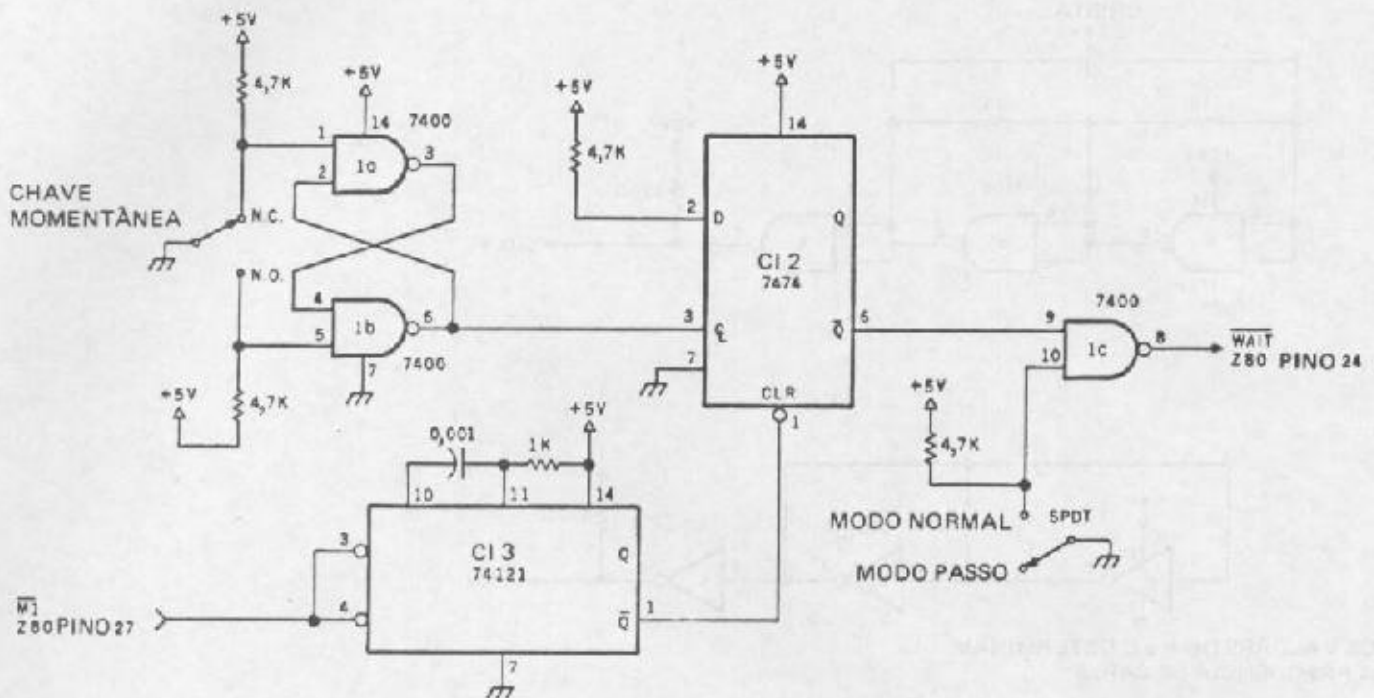



Figura 4.5 Circuito de passo a passo de instrução.

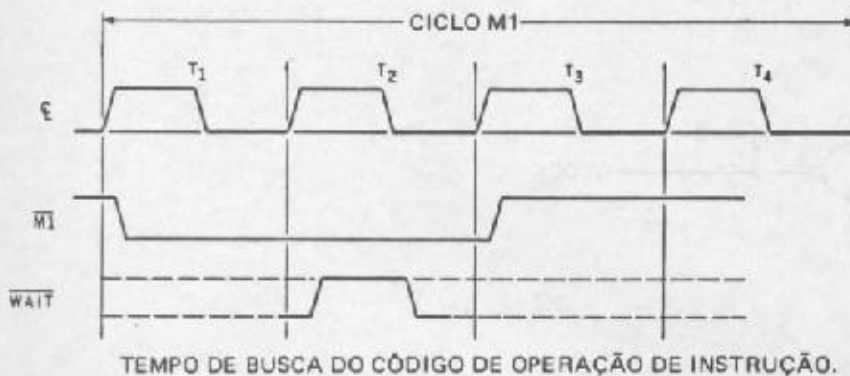



Figura 4.6 Diagrama de tempos de busca de uma instrução.

Quando os computadores são usados em aplicações onde a presença humana não se faz presente, como por exemplo em um controlador de sinal de trânsito, o rearme manual não pode ser usado, em vez disso o rearme automático deverá ser usado. A figura 4.8b mostra um circuito de rearme automático. Quando o computador é ligado, o capacitor de 10 μF estará completamente descarregado. O nível lógico zero resultante na entrada do 7404 pino 1 será mantido por aproximadamente 50 ms após a fonte de +5V ter-se estabilizado.

A razão de carga do capacitor fará com que um nível lógico 0 (zero), que é uma condição de rearme, atue no computador até que a entrada atinja o nível aproximado de 2V (nível lógico TTL de 1).

O rearme automático e manual estão combinados na figura 4.9. Este circuito permite ao computador começar a execução de um programa imediatamente após ser ligado. O programa pode ser parado e reiniciado apertando o botão de rearme. Componentes ligeiramente diferentes e funções adicionais estão incluídos neste circuito. Quando o equipamento for desligado, o uso de um diodo para descarregar o

capacitor assegura que um pulso será aplicado se a força voltar repentinamente. Como as variações de tensão da rede são geralmente rápidas, a razão de descarga do capacitor tem de ser rápida o suficiente para que não se deixe de gerar um pulso de rearme. Embora este circuito não seja necessário para o funcionamento inicial do PAZ, será de grande ajuda nas expansões, que serão mostradas posteriormente. Para que possamos sincronizar o processador central com os periféricos, estes deverão estar ligados ao mesmo sinal de saída deste circuito.

Figura 4.7 Circuitos de mostrador com LED e ponta de prova lógica.
a) Circuitos indicadores de nível que podem ser ligados às barras de endereço e dados.
b) Ponta de prova lógica simples.

Figura 4.8 Circuito de rearme.
a) Circuito de rearme normal.
b) Circuito de rearme automático.

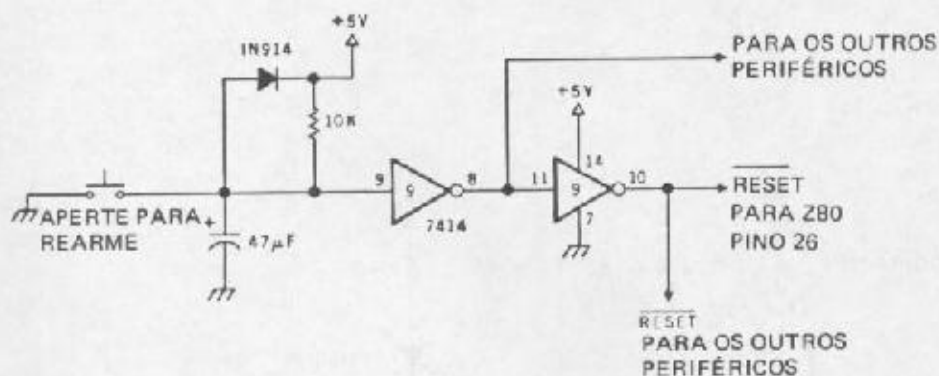



Figura 4.9 Circuito que combina as funções de rearme manual e automático.

C. Buferização das Barras de Endereço e Controle

O Z80 tem a capacidade de endereçar diretamente 65.536 (geralmente chamado 64K) palavras de memória e 256 portas de entrada e saída. Como o microprocessador é um componente binário, é natural que o seu endereçamento seja binário. Existem 16 linhas de endereço chamadas A0 a A15. A A0 é o bit menos significativo (LSB) e a A15 é o bit mais significativo (MSB). Os níveis lógicos nesta barra não são arbitrários. A parte de controle do processador central coloca o programa para a próxima instrução a ser executada, e durante o ciclo de busca coloca o conteúdo deste contador na barra de endereços. Durante as instruções de entrada/saída, ciclos adicionais são inseridos e o endereço é colocado nos 8 bits menos significativos da barra (A0 à A7). Como esta barra tem de excitar um grande número de componentes, a barra de endereços tem de ter uma capacidade grande de fornecer corrente. O Z80 por si pode fornecer 1,8 mA máximo ou uma carga TTL nos seus pinos de saída. Isto não representa problema se o projetista usar memórias e periféricos de baixa potência. Só que estes componentes são caros. Usar integrados de baixa densidade de integração e componentes TTL para funções de decodificação é mais barato, mas requer maior potência da barra de endereços. A tabela a seguir mostra a corrente de entrada de alguns componentes.

Componente	Pior caso de corrente na entrada
Padrão TTL (7404, 7442, etc)	1,6 mA
Baixa potência TTL (74LS04, etc)	0,18 mA
2708 (1K × 8 EPROM)	10 µA
2114 (1K × 4 Memória programável)	10 µA
2716 (2K × 8 EPROM)	10 µA
2102 (1K × 1 Memória programável)	10 µA
8212 (8-bit latch)	0,25 mA
8T97 (6-bit driver)	1,0 mA

É fácil de ver que os componentes TTL é que realmente consomem corrente. Componentes de baixa potência do tipo Schottky podem ser usados no computador PAZ. Eles reduzem o consumo com apenas um pequeno aumento do preço. Mas se for usado, deverá ser em todo o PAZ.

A carga apresentada pela memória, especialmente com os 2K básicos do PAZ, é insignificante. Com os 1,8 mA que podem ser fornecidos pelo Z80 nós podemos usar o TTL de baixa potência para a decodificação da memória e da entrada/saída, porém temos de limitar o fanout (número total de conexão de entradas) em cada linha de endereço em 9 entradas TTL de baixa potência. Isto é suficiente para o PAZ básico e provavelmente seria um procedimento aceitável, mas não é recomendável.

Da primeira vez que o usuário colocasse a ponta de prova lógica (figura 4.7b) em uma linha de endereço não buferizada poderia danificar o computador. A carga apresentada pela ponta de prova assim como pelos outros circuitos irá exceder a capacidade da barra de endereços. É importante que os componentes usados para monitorar a barra não impeçam o computador de funcionar. Para se solucionar o problema a melhor solução é colocarmos buffers para que a capacidade de carga seja aumentada. Com isto o usuário poderá, inclusive, incluir seus próprios circuitos TTL sem se preocupar mais com carga na barra. Para se conseguir esta maior carga na barra de endereços usaremos um buffer não inversor. As saídas de A0 a A15 são ligadas unicamente às entradas dos buffers. Qualquer outro componente que use linhas de endereço será ligado à saída dos buffers. A figura 4.10 mostra o diagrama e a tabela verdade do componente 8T97 (74367). Este componente de três estados é capaz de drenar até 48 mA e poderá acomodar qualquer configuração de componentes TTL e de memória que o usuário desejar. A configuração final da barra de endereços é mostrada na figura 4.11.

A função de três estados do (8T97 é controlada pelo sinal $\overline{\text{BUSAK}}$. Este sinal faz com que a barra de endereços fique sob o controle de um componente externo durante os acessos diretos à memória (ADM).

DIS_4	DIS_2	DATA INPUT	OUTPUT
0	0	0	0
0	0	1	1
X	1	X	HIGH Z
1	X	X	HIGH Z

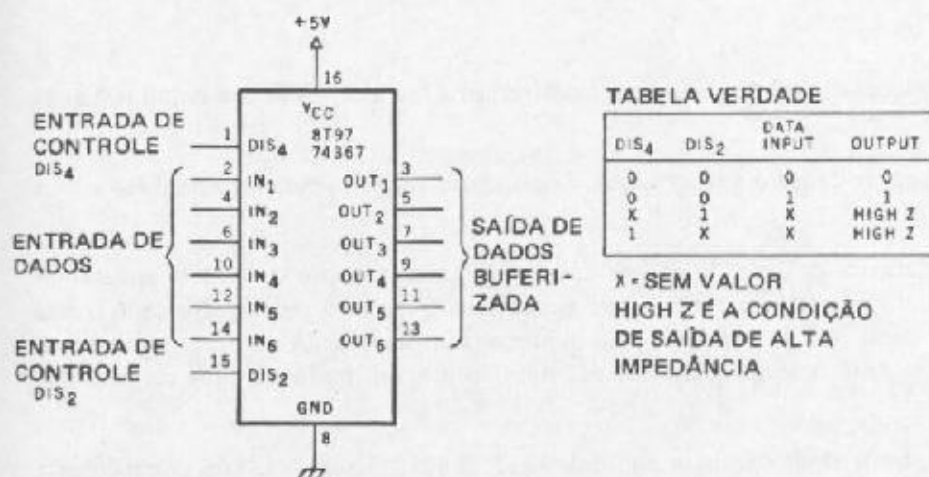



Figura 4.10 Pinagem e tabela verdade do 8T97/74367.

Se não existir uma situação de ADM, o sinal $\overline{BUSAK}$ estará alto e o 8T97 passará todas as saídas do Z80. Quando um pedido de ADM é reconhecido, o sinal de $\overline{BUSAK}$ vai a baixo colocando a saída do 8T97 no modo de alta impedância. Esta facilidade permite que se possa escrever e ler diretamente da memória por um componente externo e é geralmente reservada para operações de alta velocidade que sejam mais rápidas que o processador central possa suportar.

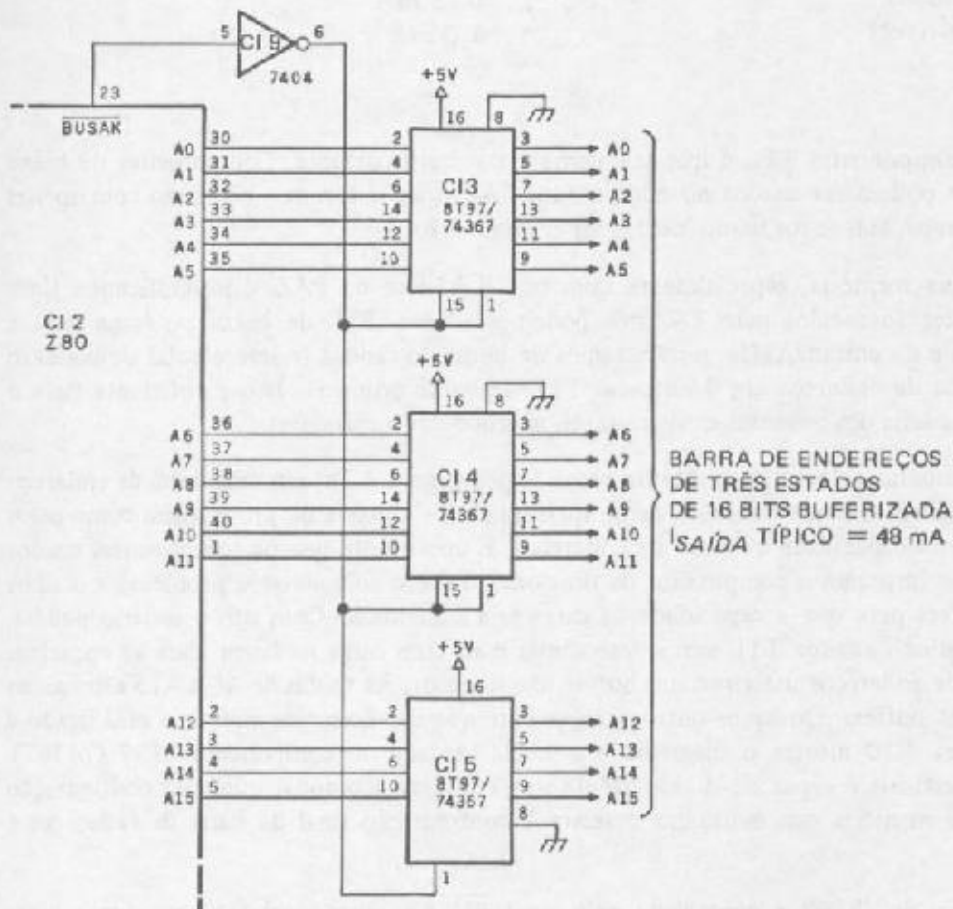



Figura 4.11 Configuração final da barra de endereços buferizada.

D. Barra de Dados e Controle

A quarta e última área de ligações diretas ao processador central são a barra de dados e as linhas restantes da barra de controle.

A razão para buferizar a barra de dados é a mesma que a de endereços; com uma pequena diferença: a barra de dados é bidirecional.

Uma barra bidirecional significa que os dados fluem em ambas as direções. Quando o Z80 está escrevendo um dado na memória, este flui do processador central para a memória. Quando o processador central está lendo um lado da memória, o dado flui da memória para o processador central. A natureza bidirecional da barra de dados requer que os buffers sejam bidirecionais internamente, ou ligados de uma maneira que executem a mesma função.

Uma maneira de se fazer um buffer bidirecional é usar dois 8212. O 8212 (figura 4.12) foi originalmente idealizado e produzido pela INTEL como um latch de 8 bits para porta de saída ou entrada. Os dados podem passar continuamente pelo 8212 ou pode ser desligado para bloquear o fluxo; encaixa perfeitamente nesta aplicação, pois apresenta saída de três estados.

Dois 8212 (figura 4.13) são montados em direção oposta um ao outro. O circuito integrado CI 6 dirige os dados do processador central para a memória e o CI 7 dirige os dados para o Z80. O controle do fluxo é dado pelo sinal de leitura $\overline{RD}$. O sinal $\overline{RD}$ está normalmente baixo, exceto durante as operações de escrita. Isto faz com que o CI6 esteja desligado e o CI7 ligado, o que permite os dados da memória ou E/S chegarem ao processador central. Quando o sinal RD foi para alto durante uma operação de escrita, o processo é inverso, o CI6 é ligado, o CI7 desligado ou E/S. Nós estamos assumindo que quando o processador central não está escrevendo, ele está lendo. Embora não sendo exatamente verdade, o conceito funciona bem na prática. A ligação dos dois 8212 é mostrado na figura 4.14.

Não é absolutamente necessário usar o 8212 para executar esta função. O 8T97 ou 74367 funcionam igualmente bem, mas usam quatro integrados. Se você não se importar com fiação extra e tiver 8T97 extras, eles podem ser fiados como ilustrado na figura 4.15.

As ligações finais ao processador central a serem discutidas são as da barra de controle, mostradas na figura 4.16. Elas coordenam periféricos, dados e endereços para dentro e fora do processador central. Cada um destes sinais foi discutido rapidamente na pinagem do Z80.

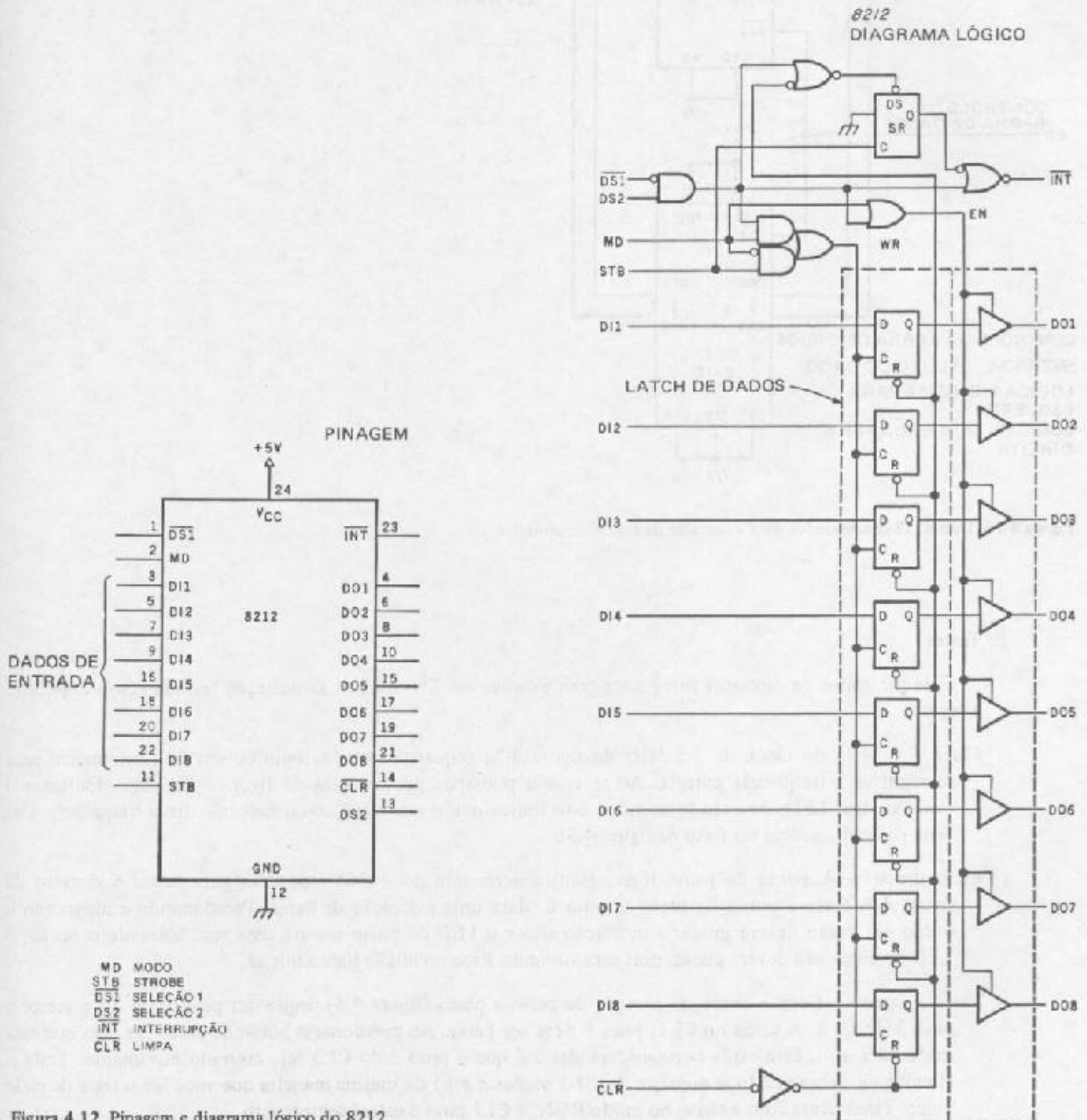



Figura 4.12 Pinagem e diagrama lógico do 8212.

Os sinais de entrada de controle que não usados são colocados em alto para evitar disparos falsos. As linhas de saída são buferizadas pelos mesmos motivos da barra de endereços.

Estas áreas discutidas mais adiante estão combinadas em um único diagrama de blocos (figura 4.7) chamado diagrama de barras do Z80 e controle.

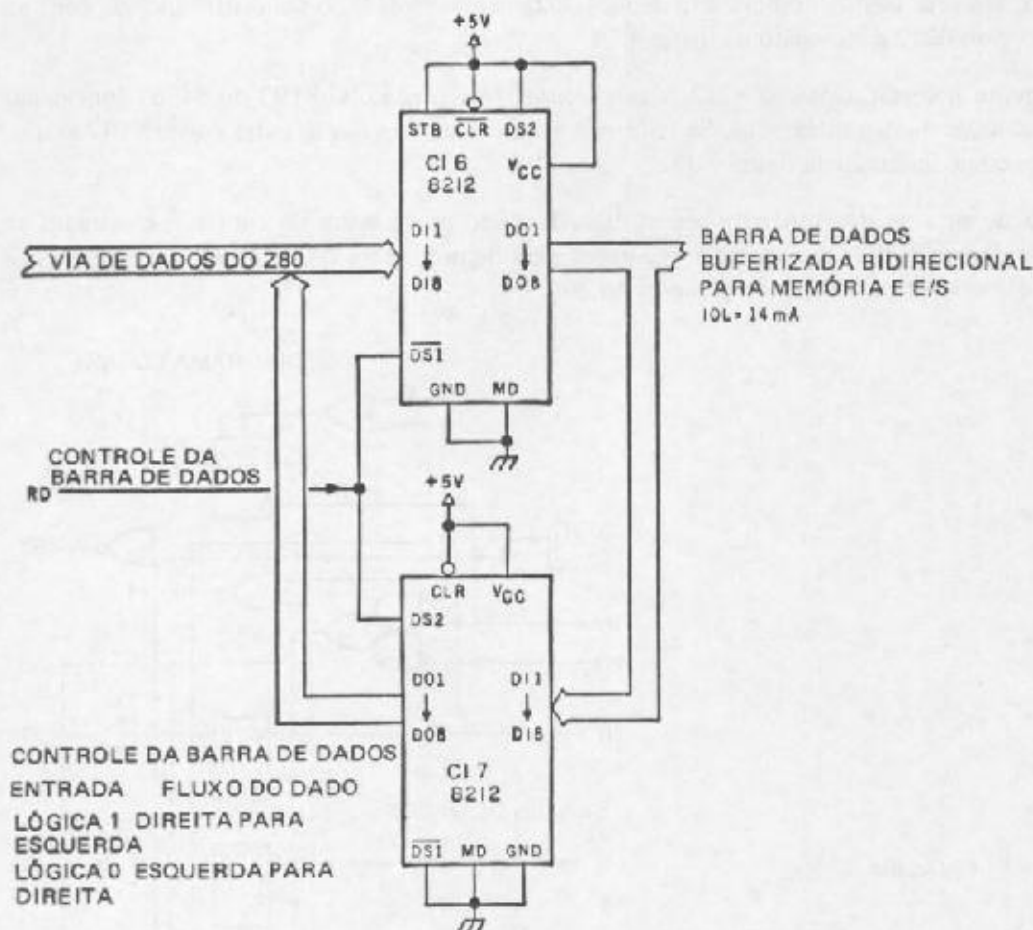



Figura 4.13 Dois 8212 configurados para a barra de dados bidirecional.

E. **Testes**

Coloque todos os circuitos integrados com exceção do Z80 e ligue. Cada seção testada como explicado a seguir.

Clock – O teste do clock de 2,5 MHz da figura 4.3a requer um osciloscópio ou um freqüencímetro para se registrar a freqüência correta. Ao se usar a ponta de prova lógica da figura 4.7b para monitorar o clock os três LEDs deverão se acender. Isto indica que o clock funciona, mas não diz a frequência. Um teste parecido poderá ser feito na figura 4.3b.

Ciclo único – A ponta de prova lógica (sem o acréscimo do 7486) é perfeita para testar o circuito da figura 4.4. Com a ponta na seção C pino 8, dará uma indicação de baixo. Pressionando e mantendo o botão em baixo deverá mudar a indicação alto e o LED de pulso piscará uma vez. Soltando o botão, o LED de pulso não deverá piscar, pois está voltando à sua condição lógica inicial.

Passo a passo – Com a chave na posição de passo a passo (figura 4.5) pegue um pedaço de fio e aterre o pino 3 do CI 3. A saída no CI 1, pino 8 deve ser baixa. Ao pressionar o botão de passo faz com que esta saída vá à alto. Esta saída permanecerá alta até que o pino 3 do CI 3 seja aterrado novamente. Teste o circuito de debounce (que consiste do CI 1 seções *a* e *b*) da mesma maneira que você fez o teste de ciclo único. Finalmente com a chave no modo RUN, o CI 1 pino 8 estará sempre alto.

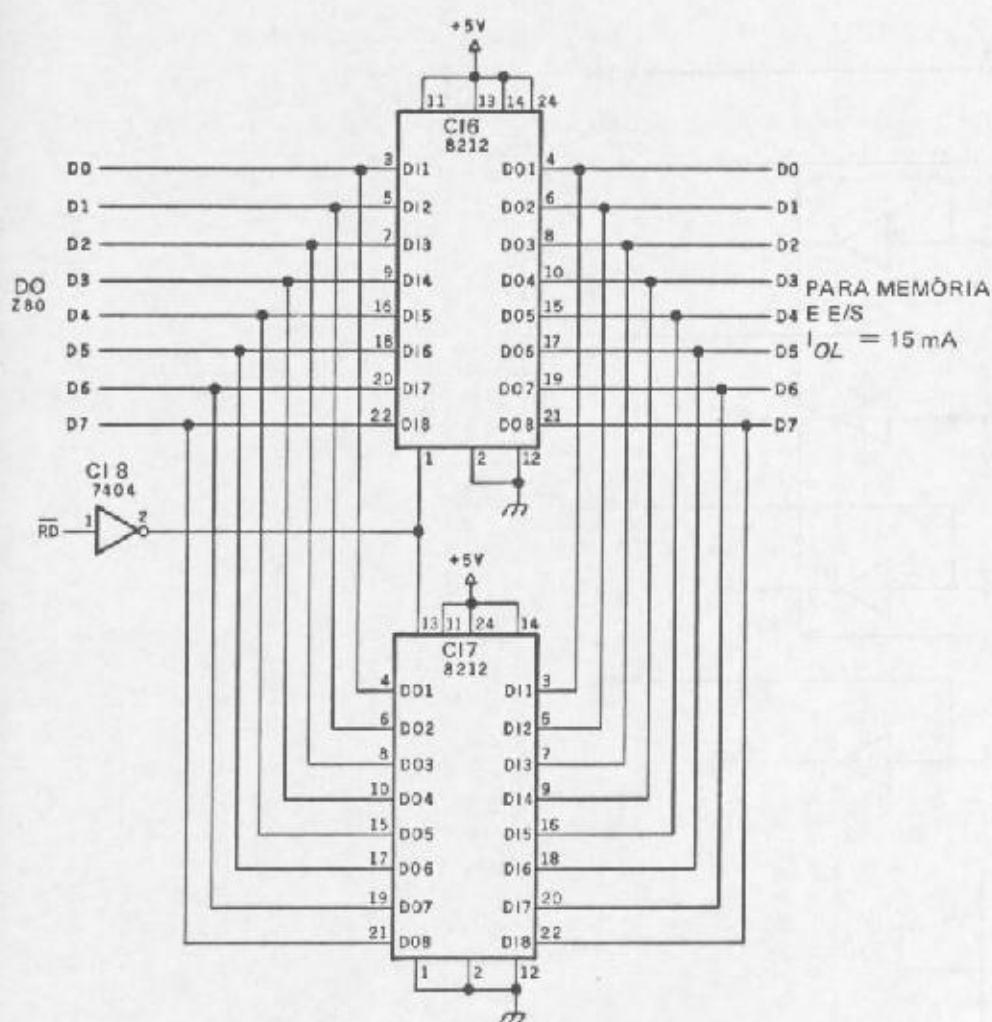



Figura 4.14 Diagrama esquemático de dois 8212 configurados para a barra de dados bidirecional.

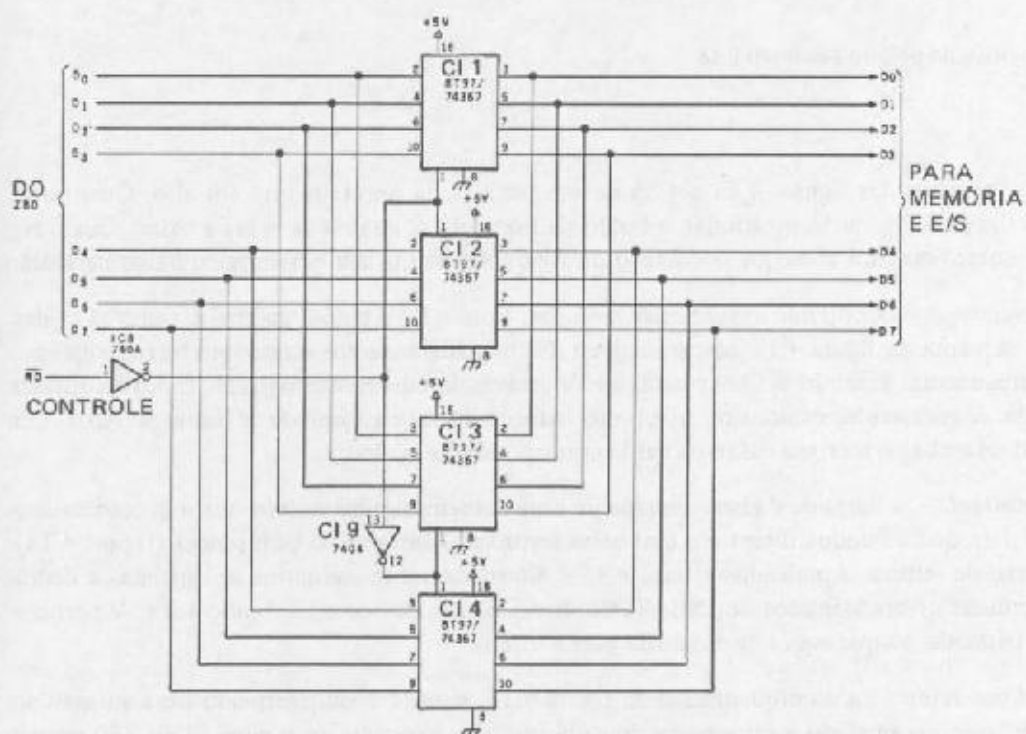



Figura 4.15 Diagrama esquemático de um buffer da barra de dados feito com 8T97.

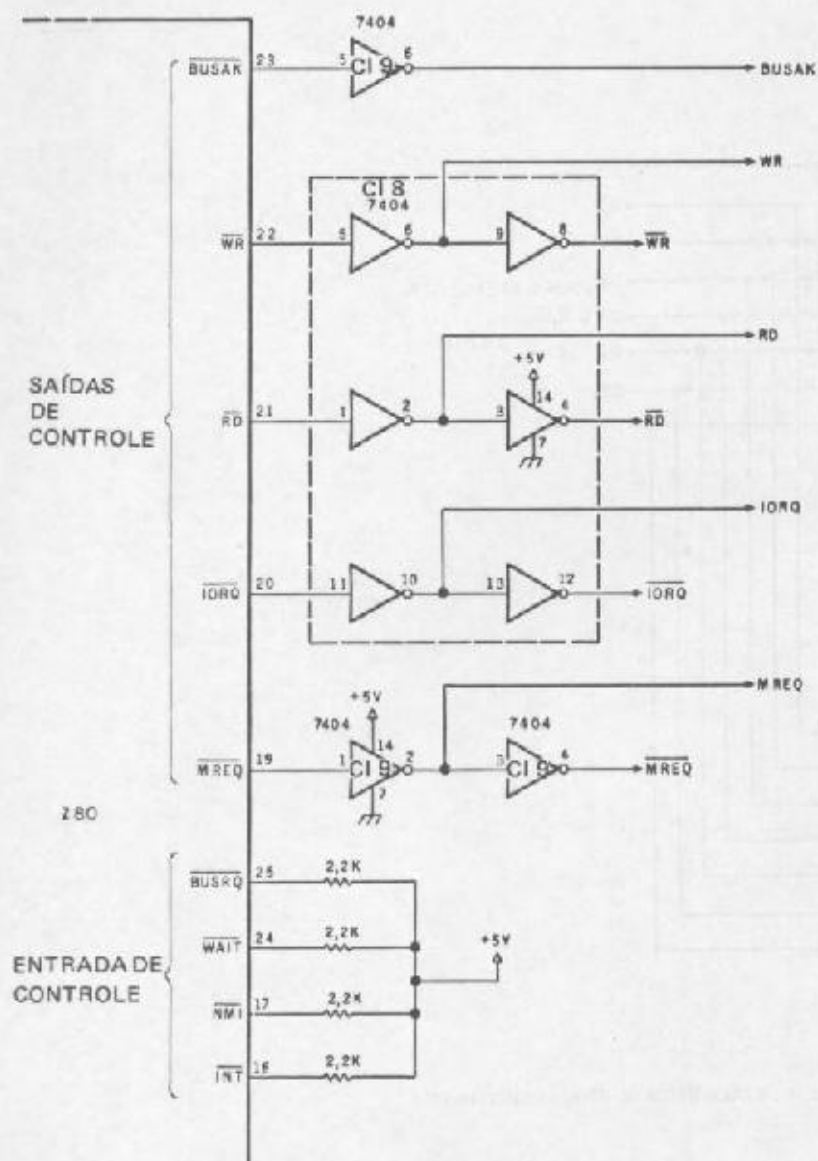



Figura 4.16 Ligações de sinais de controle do projeto básico do PAZ.

Rearme ao ligar – Os circuitos das figuras 4.8a e 4.8b devem ter a saída normalmente em alto. Quando se ligar o circuito da figura 4.8b, ou se pressionar o botão da figura 4.8a, a saída deve ser a baixo. Qualquer uma das situações anteriores fará com que o circuito da figura 4.9 tenha um nível lógico baixo na saída.

Buffers da barra de endereço – O Z80 não deverá estar no lugar! Com o CI 9 pino 5 aterrado, todas as saídas dos CI 3, 4, 5 no esquema da figura 4.11 devem ser altas. Na verdade as saídas estarão no terceiro estado, ou seja, em alta impedância. Ligando o CI 9 pino 5 à +5V através de um resistor de 2,2K todos os buffers serão ligados. Todas as suas saídas estarão no nível alto. Aterrando sucessivamente as linhas de A0 à A15 no conetor do Z80 deverá aparecer um baixo na saída correspondente ao buffer.

Barra de dados bidirecional – A barra de dados é testada de uma maneira similar exceto que o procedimento é feito duas vezes para que os dados fluam em ambos os sentidos. Aterrando o CI 8 pino 1 (figura 4.14) simula uma condição de leitura. Aplicando-se terra e +5V alternadamente aos pinos de entrada de dados do CI 6 deverá produzir níveis idênticos de D01 à D08 do CI 6. Ligando-se o CI 8 pino 1 à +5V permite uma transferência parecida, só que agora da esquerda para a direita.

Barra de controle – Com referência ao esquemático da figura 4.16, o teste é simplesmente uma questão de se aplicar um nível lógico conhecido a uma porta de cada vez. Por exemplo, se o pino 19 do Z80 estiver com um nível lógico baixo, o pino 2 do CI 9 estará com o nível lógico alto e o CI 9 pino 4 estará baixo. A cada seção do inversor pelo qual o sinal passa, inverte o sinal.

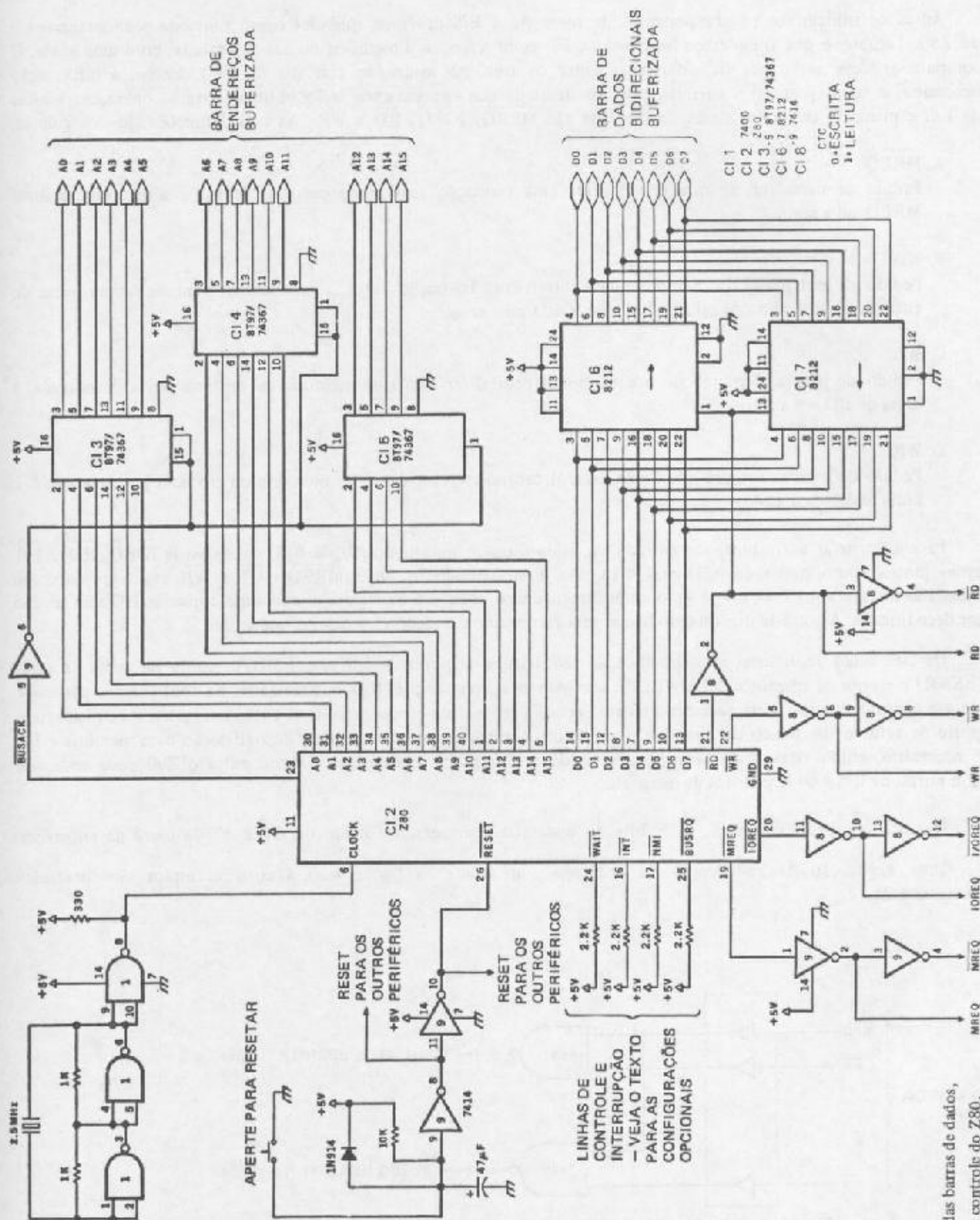



Figura 4.17 Diagrama das barras de dados, endereço e controle do Z80.

II. Decodificação de Memória e E/S

Antes de utilizarmos os componentes de memória e E/S devemos aprender como funciona o endereçamento do Z80. Lembre-se que o endereço hexadecimal FF pode referir-se à memória ou a uma entrada, ou a uma saída. O computador deve ser capaz de diferenciar entre os três. As saídas de controle do Z80 contém a informação necessária, e se as juntarmos corretamente, os sinais de que necessitamos serão obtidos. Para as operações básicas de E/S e memória os quatro sinais de interesse são $\overline{MERQ}$, $\overline{IORQ}$, $\overline{RD}$ e $\overline{WR}$. As suas definições são as seguintes:

A. $\overline{MREQ}$
Pedido de memória. Sempre que ocorrer uma transação entre o processador central e a memória, a linha MREQ vai a zero.

B. $\overline{IORQ}$
Pedido de entrada/saída. Sempre que ocorrer uma transação entre o processador central e uma porta de entrada ou uma porta de saída, a linha de $\overline{IORQ}$ vai a zero.

C. $\overline{RD}$
Pedido de leitura. Sempre que o processador central ler dados da memória ou de uma porta de entrada, a linha de RD vai a zero.

D. $\overline{WR}$
Pedido de escrita. Sempre que o processador central escrever dados na memória ou em uma porta de saída, a linha de $\overline{WR}$ vai a zero.

Para diferenciar entre porta de entrada ou saída durante uma operação de E/S, os sinais de $\overline{IORQ}$, $\overline{RD}$ e $\overline{WR}$ estão juntos como mostrado na figura 4.18. Do mesmo modo os sinais MREQ, RD e WR estão juntos como mostrado na figura 4.19. Como já visto anteriormente uma condição de leitura de memória como de E/S não precisa ser decodificada. Assume-se que quando não se está escrevendo o processador central está lendo.

Os três sinais resultantes da decodificação são: leitura de porta de entrada ($\overline{ESRD}$), escrita na porta de saída (ESWR) e escrita de memória (MEMWR). Se somente estas três funções fossem necessárias na configuração particular do seu computador não seria necessário outras decodificações. Este computador teria então uma porta de entrada, uma porta de saída e um banco de memória. Para solucionarmos este problema, uma decodificação para memória e E/S é necessário, então estes sinais servirão a mais de um dispositivo. Com um circuito extra o Z80 pode endereçar 256 portas de E/S e 64 K palavras de memória.

Durante um pedido de E/S, os 8 bits do endereço aparecem nas linhas de A0 à A7 da barra de endereços.

Uma explicação da codificação de endereço é mostrada na figura 4.20. Outros exemplos são mostrados na figura 4.21.

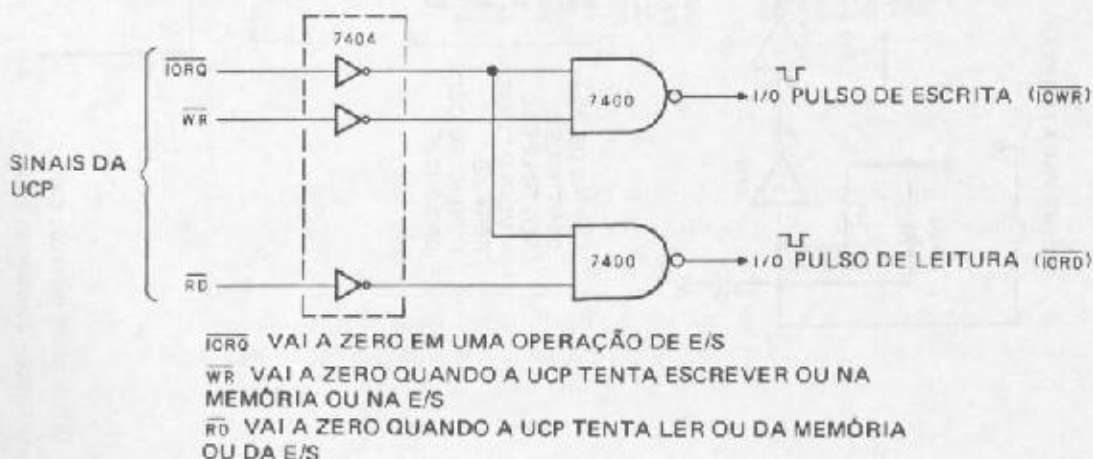



Figura 4.18 Decodificação de E/S de leitura e escrita.

Usando esta informação, se uma instrução estivesse endereçando a porta de saída 7, o circuito da figura 4.22 poderia ser usado. Quando um código de 007 em octal (07 em hexadecimal ou 00000111 em binário) aparece nas linhas de endereço, com o sinal de ESWR, o sinal presente na barra de dados seria armazenado em um registrador de 8 bits como dado de saída.

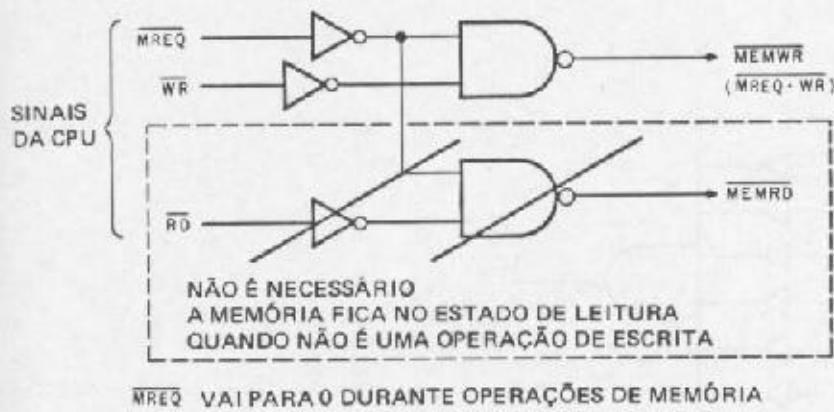



Figura 4.19 Decodificação de leitura e escrita de memória.

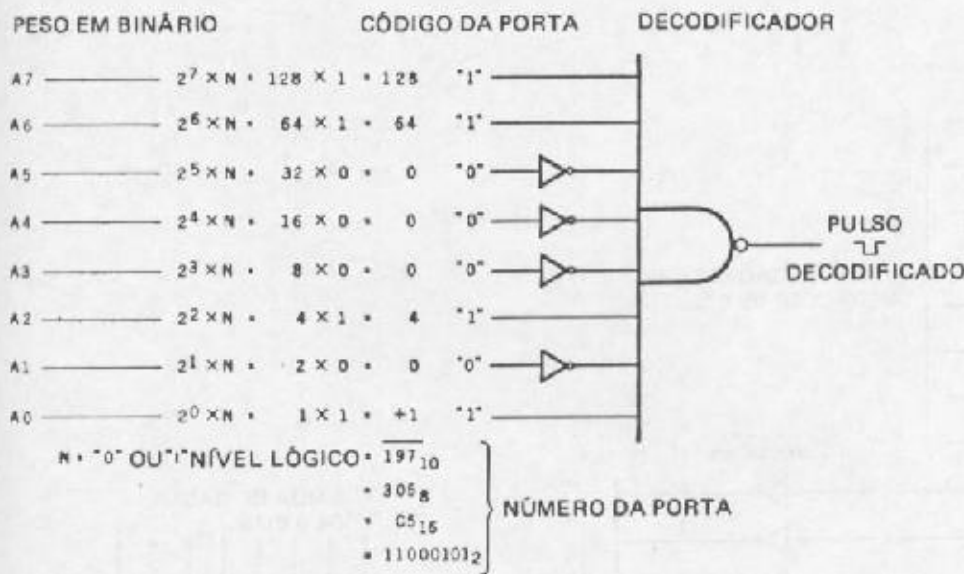



Figura 4.20 Uma explicação dos códigos de endereço de E/S.

Decodificação de E/S

Naturalmente o PAZ precisa de mais de uma porta de saída, mesmo com um sistema básico. Em realidade se ele for expandido para incluir alguns dos periféricos adicionais, irá precisar de 6 a 8 portas. A decodificação destas portas adicionais não irá requerer 8 circuitos separados como na figura 4.20 ou 4.21.

Se incorporarmos um demultiplexador de 4 para 10 linhas no projeto, poderemos conseguir oito portas. O circuito da figura 4.23 pode ser usado tanto para entrada como para saída e é endereçado de 000 octal a 007 octal. Ele funciona selecionando uma das duas saídas não usadas (CI 3 pinos 9 ou 10) quando o endereço não corresponde ao decodificador.

As linhas de A3 a A7 devem ser tratadas da mesma maneira como apresentado na figura 4.20, mas A0 a A2 servem como entrada para o 7442. Esses 3 bits designarão uma das 8 linhas possíveis quando a saída do CI 1 for baixa.

Se duplicássemos este circuito para obtermos 8 portas separadas de entrada e saída (endereçadas de 000 a 007) precisaríamos de 7 integrados. O número de integrados pode ser reduzido a 3, o circuito que faz isto é mostrado na figura 4.24. Como na figura 4.23 este circuito decodifica os endereços de 000 octal a 007 octal.

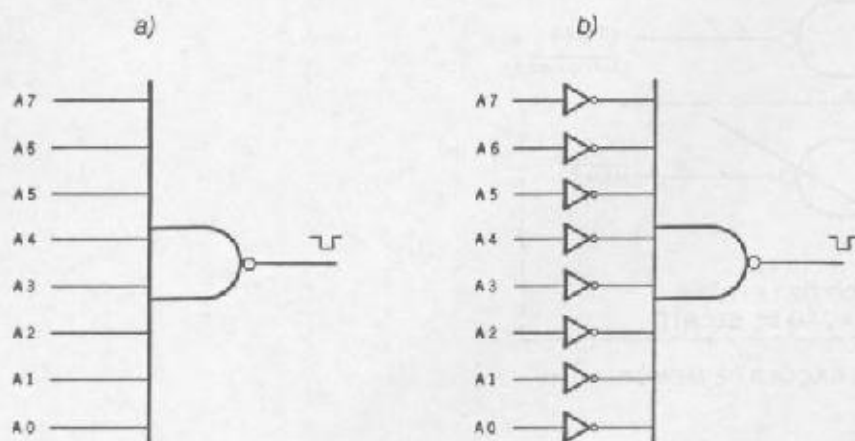



Figura 4.21 Lógica de decodificação de endereço.
a) Endereço FF_{16}.
b) Endereço OO_{16}.

BARRA DE ENDEREÇOS
A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8
NÃO SÃO USADAS PARA OPERAÇÕES DE E/S
A_7 L
A_6 L
A_5 L
A_4 L
A_3 L
A_2 H
A_1 H
A_0 H
DEVICE NO. "7" DECODER
$\overline{IOWR}$
SAÍDA DE DADOS DE 8 BITS
REGISTRADOR DE 8 BITS
LATCH
$DI_1 - DI_8$
BARRA DE DADOS DE 8 BITS
NOTA: O FLUXO DE DADOS E DA UCP PARA A PORTA DE SAÍDA DURANTE AS OPERAÇÕES DE $\overline{ESWR}$

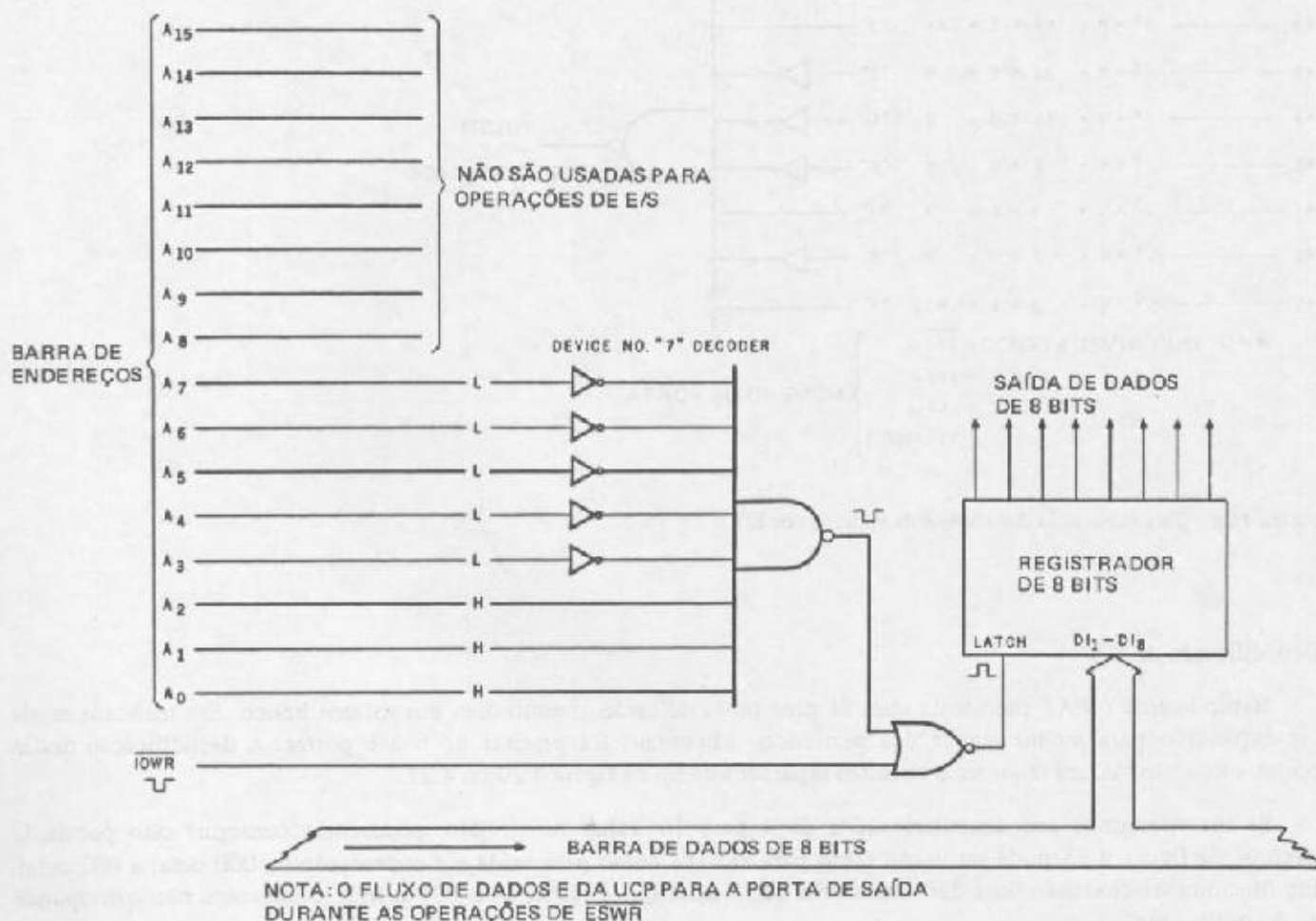



Figura 4.22 Decodificação para uma porta de saída de 8 bits. O endereço desta decodificação é 0078.

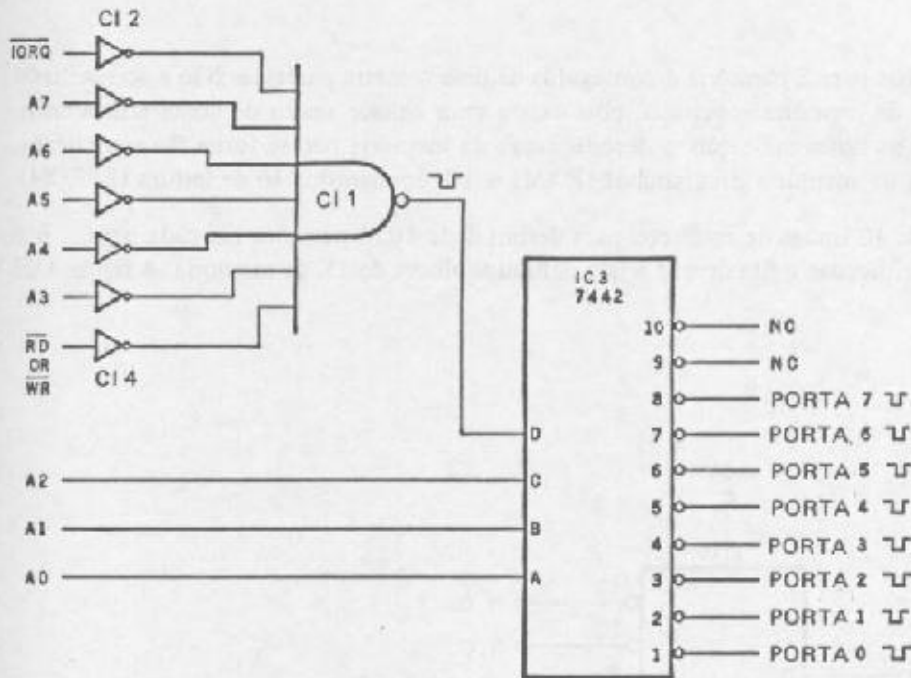



Figura 4.23 Método de decodificação de endereços que decodifica 8 linhas de endereço.

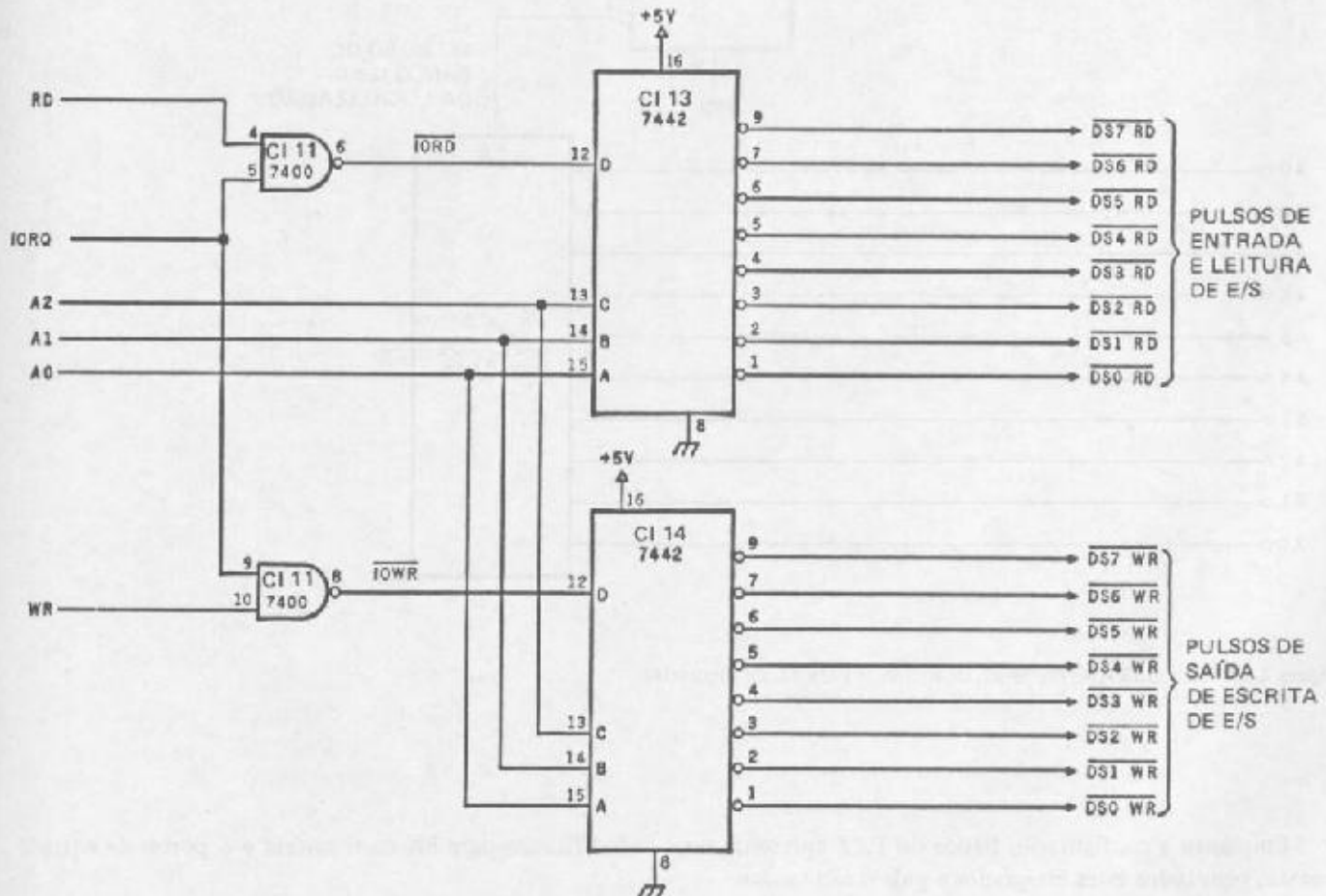



Figura 4.24 Método de decodificação com um número reduzido de componentes.

Decodificação de Memória

A decodificação da barra de endereços para a memória é conseguida de uma maneira parecida. Não é aconselhável entretanto fazermos uma decodificação de memória repetitiva, pois existe uma chance maior de cometermos erros. Embora envolvidos agora com 16 linhas na nossa aplicação, a decodificação da memória não se torna tão complicada. O PAZ usa bancos de 1K X 8 palavras de memória programável (RAM) e 1K de memória só de leitura (EPROM).

Ambos os componentes precisam de 10 linhas de endereço para definir 1 de 1024 posições em cada banco. Isto deixa apenas seis linhas para serem decodificadas a fim de que sejam definidos blocos de 1K de memória. A figura 4.25 mostra como isto pode ser conseguido.

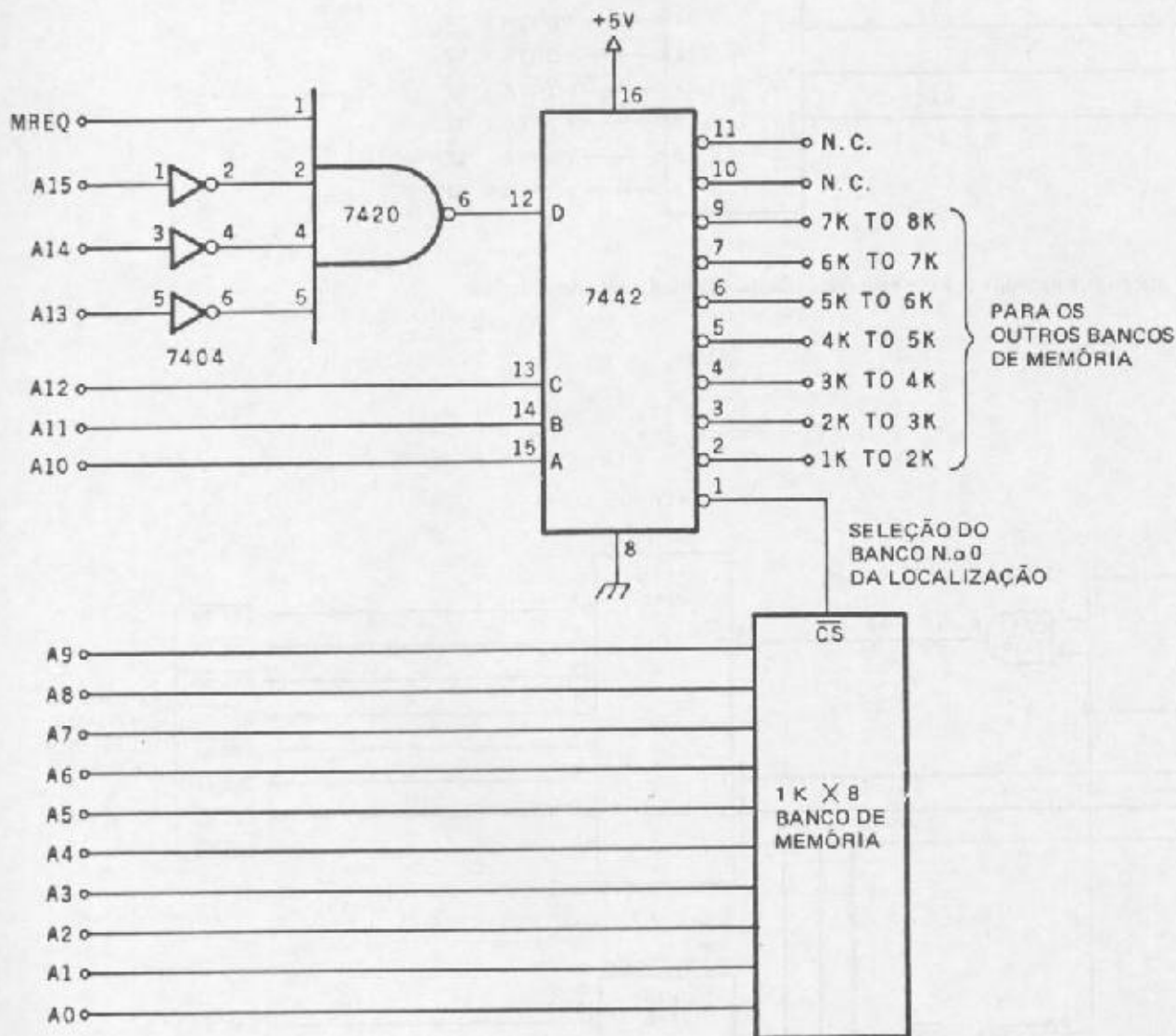



Figura 4.25 Decodificação do banco de memória para 8K de memória.

Enquanto a configuração básica do PAZ apresenta uma decodificação para 8K de memória e 8 portas de entrada e saída, nem todos estes integrados e pulsos são usados.

As linhas extras são deixadas para expansão. A figura 4.26 mostra o esquemático completo da decodificação de memória e E/S para o montador adicionar ao circuito da figura 4.17.

Testando

Após você ter adicionado os componentes da figura 4.26 aos da figura 4.17, você está pronto para testar a decodificação da memória e da E/S. Coloque os CIs, 10, 11, 12, 13 e 14, mas não coloque ainda o CI 20. Os CIs 1, 3 e 9 devem permanecer inseridos desde o teste anterior. O Z80 deve ainda ficar fora. O nível lógico de cada endereço de entrada D dos 7442 (CIs 12, 13 e 14) deve ser alto. Se retirarmos os CIs 8 e 9 (com a fonte desligada) faremos com que esta entrada mude imediatamente para um nível lógico baixo.

Em seguida, no soquete do Z80 coloque o pino 23 em nível lógico alto e os pinos 30, 31 e 32 em terra. Com os buffers da via de endereço habilitados e o endereço de A0 e A2 com 000, aparecerá um nível baixo no chip-select do endereço de carga (strobe) mais baixo. Neste caso, o pino 1 dos CIs 13 e 14 deve estar baixo e as outras linhas de estrobe devem estar altas. O decodificador do banco de memória funciona da mesma forma exceto que será através das linhas de endereço de A10 a A12.

Após este teste, coloque todos os chips exceto o Z80.

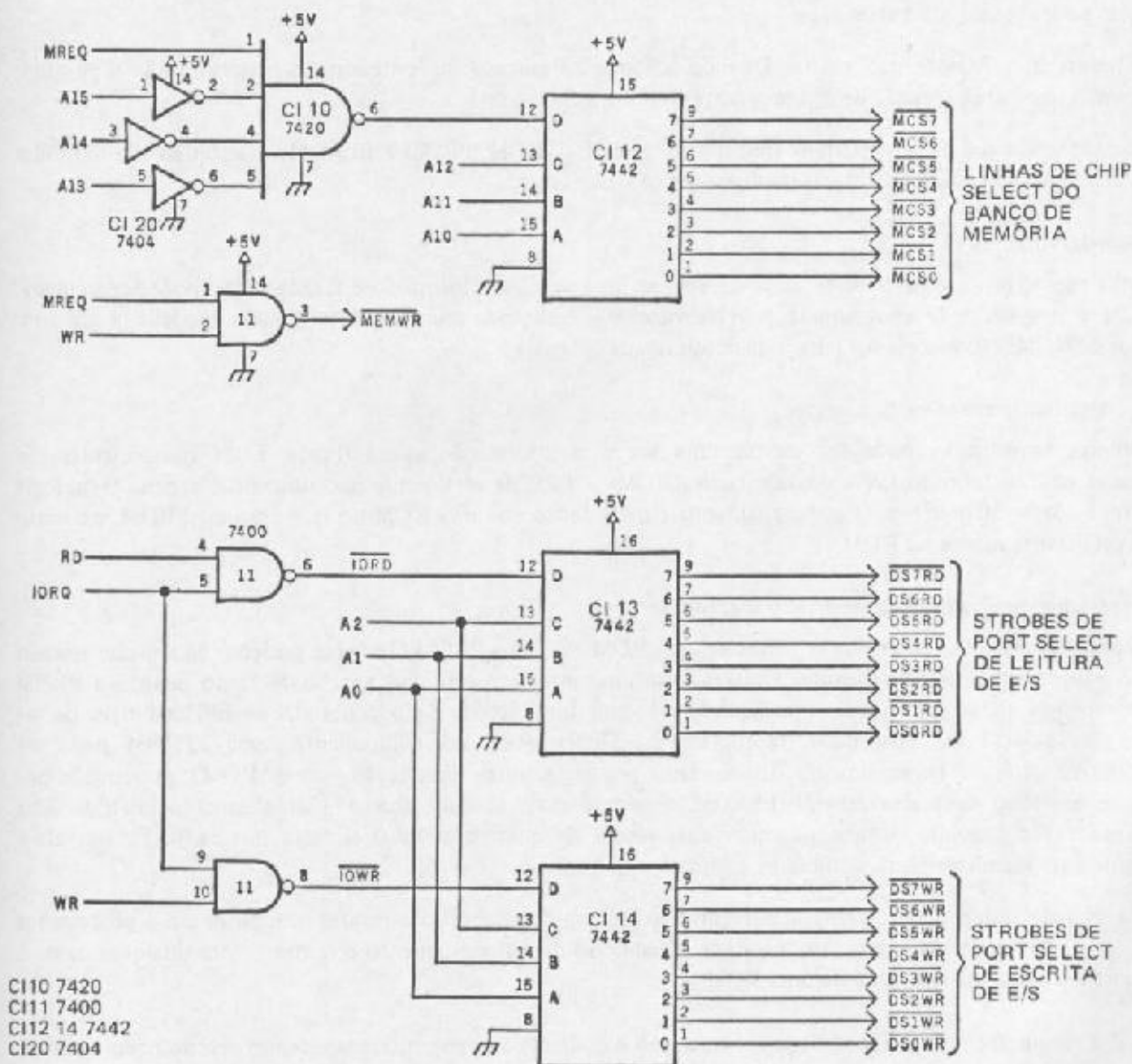



Figura 4.26 Seção de decodificação da memória e E/S.
a) Strobes dos chip-select do banco de memória.
b) Strobes dos chip-select dos dispositivos de entrada/saída.

III. Memória

Claro que em qualquer computador uma grande consideração é dada à memória. Tanto instruções quanto dados do programa devem ser armazenados e requisitados no devido tempo a fim de que o computador possa executar sua função. Apesar do processador central Z80 ter uma quantidade de registros de carga de 8 bits, estes só podem ser utilizados para manipulação temporária de dados e não podem armazenar instruções de programa. As instruções de programa devem ser armazenadas em elementos externos de memória.

A memória externa pode ser dividida genericamente em duas classes: ROM (memória de leitura exclusiva) e RAM (memória de leitura e escrita). A ROM é utilizada para armazenar dados ou passos de programa específicos e imutáveis. Os conteúdos dessas posições de memória são considerados permanente e não podem ser facilmente mudados. Por outro lado, a memória de leitura/escrita é usada para armazenar dados que podem ser mudados durante a operação do computador. Para ambos os tipos de memória, a função final é a mesma: prover, quando requisitada, ou uma instrução para execução ou uma posição onde o dado pode ser armazenado.

Memória de Leitura Exclusiva

A ROM (read-only memory) é uma parte importante do computador, funciona como uma memória cujos conteúdos, uma vez gravados por técnicas especiais de programação, não podem ser alterados pelo processador central. Existem poucas exceções a esta regra.

Pela sua natureza, a ROM é não volátil. Quando a fonte é desligada, o conteúdo do programa não é perdido; permitindo assim a imediata execução de seu programa ao religarmos a fonte.

Dentro da categoria das ROMs existem três subcategorias — ROM, PROM e EPROM — as quais são definidas mais pela sua utilização e aplicação do que o implícito pelos seus nomes.

ROM (read-only memory)

Esta é uma memória na qual pode-se escrever apenas uma vez. A informação é fixada e não pode ser mudada. Uma ROM é normalmente programada pelo fabricante e é vendida com um determinado modelo já gravado. Estes tipos de ROMs são planejados para uma determinada clientela.

PROM (programmable read-only memory)

Esta memória também só pode ser escrita uma vez e a informação estará fixada. Estes componentes são programados pelo usuário ao invés do fabricante. ROMs e PROMs geralmente não utilizam a mesma tecnologia de construção de semicondutor. O armazenamento é mais denso em uma ROM do que em uma PROM, e o custo por bit é geralmente menor na ROM.

EPROM (erasable-programmable read-only memory)

Esse componente combina as melhores partes de uma ROM e de uma PROM. Todas as posições de armazenamento não estão programadas pelo fabricante. Utilizando-se uma interface especial, a EPROM tanto quanto a PROM podem ser programadas pelo usuário, utilizando-as como uma ROM. Se o conteúdo da EPROM tiver de ser mudado, esta poderá ser apagada e reprogramada. Dependendo do componente, uma EPROM pode ser eletronicamente alterada (normalmente diferenciado por uma outra abreviação — EAROM) ou apagada por ultravioleta, algumas vezes chamada UVEPROM. Porém é mais comum chamá-la simplesmente EPROM. Elas são facilmente reconhecidas porque possuem uma janela de quartzo sobre o circuito integrado. Esta janela é transparente para luz ultravioleta facilitando, assim, seu apagamento.

Para cada posição independentemente endereçável existe um bit específico armazenado. Somente o processador pode determinar se este é uma instrução ou um dado. O método de armazenamento é o mesmo em qualquer caso. A figura 4.27 mostra o diagrama em bloco de uma ROM.

Uma ROM é simplesmente um bloco lógico o qual, sob o controle do programa, fornece um determinado modelo. A figura 4.28 é uma memória de leitura exclusiva de 3 bits. Quando a chave SW1 está fechada (posição que poderia ser tomada quando o processador central necessitasse da informação), o código de 3 bits "101" aparecerá nas saídas. O diodo aterra os sinais de entrada dos inversores 7404 quando SW1 está fechada. Uma expansão para mais do que 3 bits é simplesmente um caso de adição de mais diodos, resistores e buffers. Tal circuito é chamado ROM de matriz de diodos e nesse caso seria uma ROM de 1 linha por n-bits.

Uma memória de 3 bits não é muito usada, mas o seu conceito pode ser facilmente expandido para 16 bytes através da adição de um decodificador de endereço conforme o diagrama da figura 4.29. A figura 4.30 ilustra um esquemático completo com os diodos especificamente agrupados para executar um simples programa de 9 bytes. Esse pequeno programa de teste será usado mais tarde durante a fase final.

A ROM de matriz de diodo é apresentada somente por seu valor educacional. Este não é um método que deva ser implementado no computador PAZ. Existem circuitos integrados capazes de preencher os requisitos de cada uma das três categorias, dessa forma devemos analisar nossas necessidades um pouco mais de perto.

As perguntas pertinentes são: tamanho da memória, custo e facilidade da programação. O tamanho de uma ROM é determinado pelo usuário. Qual o esforço que o usuário desejará dispender para fazer com que o computador execute um programa específico, ao ser ligado? Nosso computador não tem painel frontal nem bancos de chaves de endereços e dados que possam ser transformadas em instruções. Esse é o caso, o PAZ deve ter um programa que execute imediatamente (quando for ligado ou quando o reset for ativado), e que permita o processador central comunicar-se com os seus periféricos colocando-o em um modo que possa ser diretamente programado através destes periféricos. Uma vez ligado, um programa simples de 50 a 100 bytes pode ser escrito, o qual facilita um carregamento da memória através do teclado. Mas será que necessitaremos entrar com um programa grande na memória? Teremos de entrar com tudo isto através do teclado?

Dispositivos de entrada de dados de alta velocidade também podem ser acoplados através de uma interface serial. Este pode ser adicionado a uma outra memória de 100 ou 200 bytes. Uma outra consideração é a necessidade para alguns operadores de um display (mostrador) de endereços e dados para facilitar o desenvolvimento de programas.

Afinal, para incorporarmos todas as funções necessárias para um sistema em um único cartão, poderemos ter facilmente a necessidade de armazenamento de 500 a 1000 bytes em ROM. Muitos sistemas utilizam uma memória ROM de 64 a 256 bytes para armazenar o programa inicializador (bootstrap). O bootstrap é um programa que coordena o mínimo de periféricos necessários para que se possa carregar no computador um programa maior. Em muitos dos sistemas de computadores pessoais, este bootstrap controla uma interface para cassete e o programa subseqüentemente carregado é chamado monitor.

O monitor (explicado no capítulo 6) é uma parte importante do software que requer cerca de 1K de memória. Nossa decisão recairá entre colocarmos o monitor totalmente residente em ROM (pronto para execução imediata) reduzindo para um mínimo de ROM e carregarmos o monitor a partir do teclado ou de um cassete.

Essa é uma consideração importante para quem está construindo um computador. Quando se tem uma escolha, eu penso, você deverá quase sempre optar pela solução que necessite o mínimo de acessórios e você incluirá no hardware o monitor em ROM. Seria como querer colocar a carroça na frente dos burros, utilizar-se de uma interface para cassete para se carregar o software inicial. Com o monitor em ROM, pode-se entrar com os programas do usuário via teclado sem termos de construir uma interface serial. A partir de um programa monitor, residente em 1K ROM, poderemos habilitar uma interface serial e um cassete. Outra vantagem é que o PAZ estará apto mais rapidamente para programação.

Eu sugiro que o tamanho da memória ROM seja de 1K, como visto anteriormente a ROM é programada pelo fabricante e a PROM seria muito cara se utilizada em um bloco de 1K, porém ideal para um bootstrap de 64 bytes.

A sugestão alternativa para a memória de leitura exclusiva é a utilização da EPROM que é programada pelo usuário. Uma EPROM de 1K tal qual a 2708 (ou a 2K – 2716) é a de custo ideal para o computador feito em casa. A memória EPROM Intel 2708 é a recomendada para esta aplicação. (A 2716 é uma EPROM de 2K com alimentação única de +5V.)

Figura 4.27 Diagrama bloco de uma memória de leitura exclusiva.

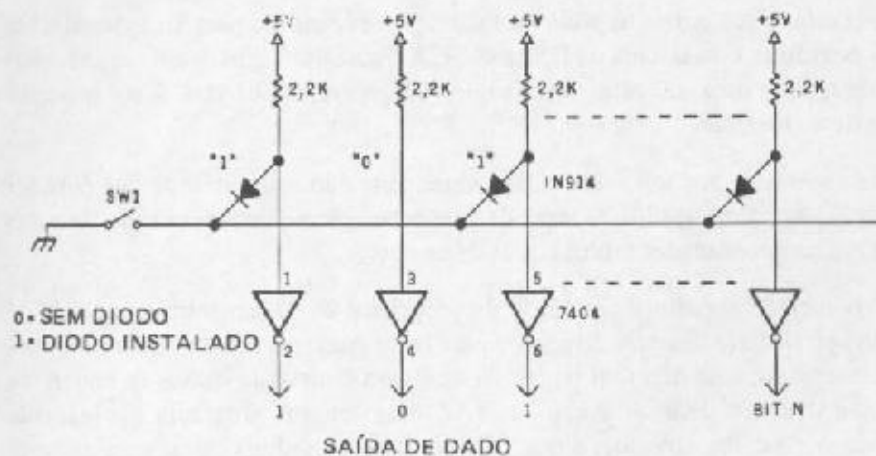



Figura 4.28 Memória de leitura exclusiva de apenas 3 bits (1 X 3 bits).

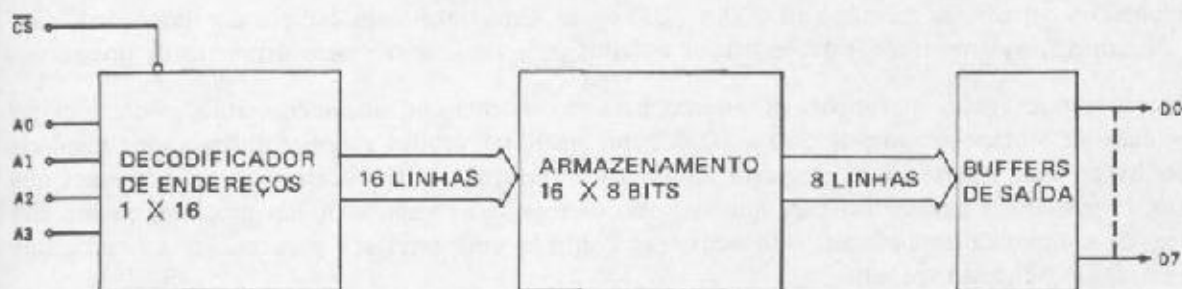



Figura 4.29 Diagrama bloco de uma ROM de 16 bytes.

EPROMs

A EPROM é a memória de leitura mais usada. Ela é utilizada como uma ROM para um período de tempo longo, apagada ocasionalmente e reprogramada quando necessária. O apagamento é permitido pela janela de quartzo transparente que cobre o substrato do chip, através de luz ultravioleta. O elemento de memória utilizado pela Intel na EPROM 2708 é um tipo de carga armazenada chamada transistor FAMOS (Floating gate Avalanche injection Metal Oxide Semiconductor). Este é similar ao transistor de efeito de campo com gate em canal *p*, porém com o gate mais baixo ou "flutuante" totalmente envolvido por um isolador de dióxido de silício. O valor 1 ou 0 armazenado na célula FAMOS é uma função da carga no gate. Uma célula carregada terá na saída um valor armazenado oposto ao de uma célula descarregada. Aplicando-se uma tensão de carga de 25V às células seletivamente endereçadas, teremos um determinado conjunto de bits que constituem o programa escrito na EPROM. Envolvida por um material isolante a carga permanecerá por anos. Quando este isolador de dióxido de silício é exposto à luz ultravioleta intensa, este torna-se condutivo e descarrega a carga do gate. O resultado é o apagamento de toda a informação programada.

Os apêndices C1 e C2 mostram a disposição dos pinos e as especificações elétricas da 2708 e 2716 respectivamente. O capítulo 7 explora vários métodos para programar e testar o chip.

Memória de leitura e escrita

A memória de leitura e escrita é justamente o que o seu nome diz. Tal memória permite que dados sejam tanto escritos como lidos de seu interior. A memória de leitura/escrita para microcomputadores é geralmente configurada a partir de circuitos semicondutores de memória programável que retém os dados somente enquanto a fonte estiver ligada.

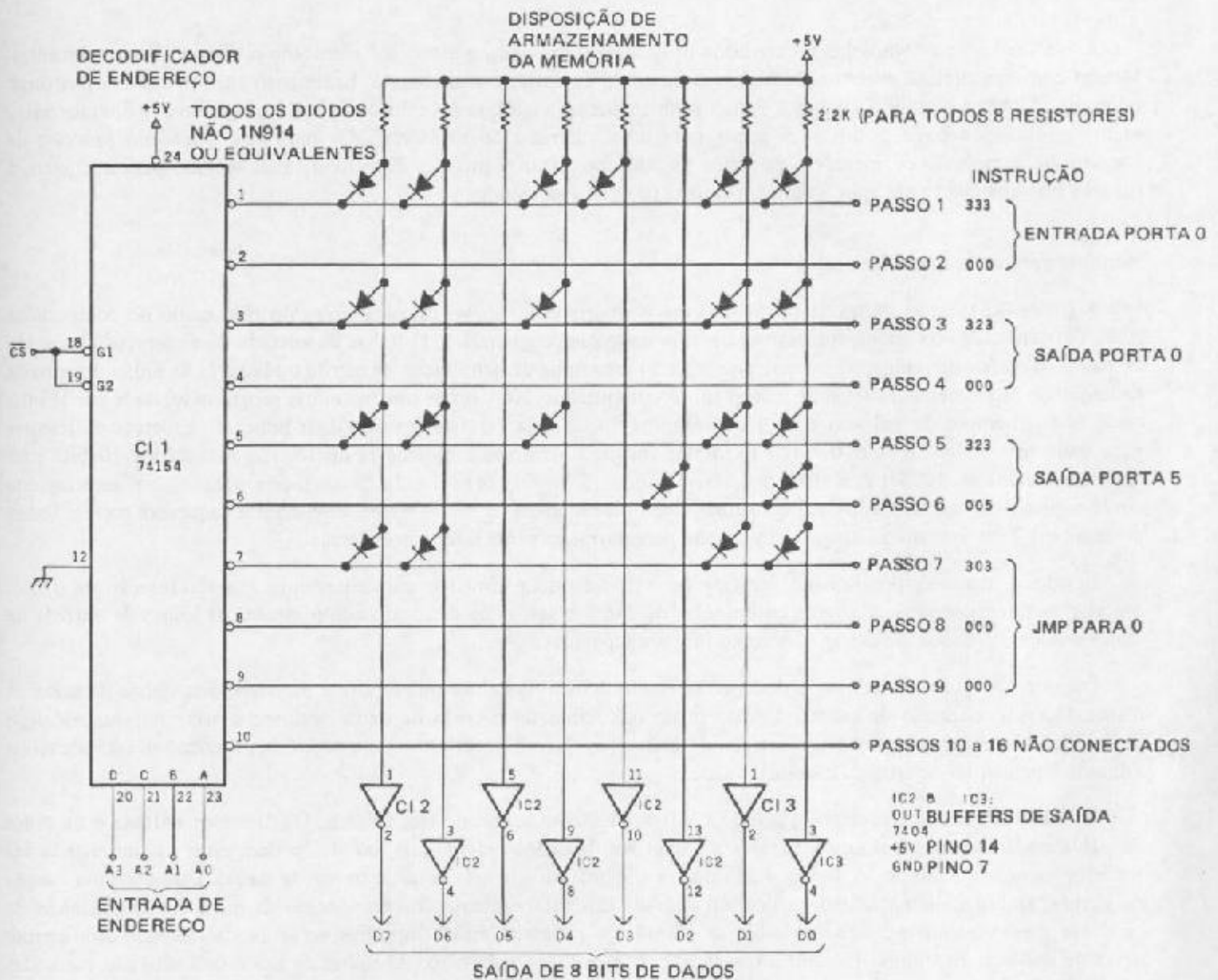



Figura 4.30 ROM de matriz de diodos com programa de teste.

A memória de leitura/escrita, que tecnicamente é uma ROM de acesso aleatório e é composta de dispositivos semicondutores passou a chamar-se de RAM (random access memory) — memória de acesso aleatório. A partir daí vamos nos referir à RAM como memória programável.

Existem duas classes de memória programável: estática e dinâmica. A memória programável estática armazena cada bit de informação em uma célula de armazenamento biestável tal qual um flip-flop. Esta informação é retida tanto tempo quanto a fonte estiver alimentando o circuito. As memórias programáveis dinâmicas possuem uma estrutura interna mais simples, tamanho menor, menos dissipação de potência, e são inerentemente mais rápidas. Elas armazenam informação como uma carga elétrica no substrato do gate de um transistor MOS. Esta carga dura somente uns poucos milissegundos e deve ser restaurada. Esta necessidade de restaurar a informação armazenada é uma das maiores distinções entre memórias programáveis estáticas e dinâmicas.

Entretanto a restauração de memória dinâmica pode ser um incômodo. O processo requer que todas as células de armazenamento sejam endereçadas pelo menos uma vez a cada poucos milissegundos (geralmente 2). Um circuito contador é geralmente incorporado para exercitar as linhas de endereço de memória quando o computador não está acessando à memória. Em muitos sistemas, a restauração (refresh) requer um circuito externo adicional. O Z80 possui este circuito dentro do chip processador central, facilitando assim a utilização de memória dinâmica. Entretanto, esta facilidade é perdida quando o Z80 é resetado; por isso torna-se necessário um circuito extra de refresh.

A escolha entre a tecnologia de memória programável dinâmica e estática é predicado do custo e da conveniência. Mesmo com um circuito externo de refresh, a memória dinâmica é mais barata. Entretanto em um sistema protótipo tal como o PAZ, a memória dinâmica é mais problemática. Desde que se tenha construído um sistema já operacional, a memória dinâmica pode muito bem servir para uma extensão de memória. Mas para este ponto do processo de construção, a inclusão de memória dinâmica poderá complicar o projeto. Este livro, mais voltado para iniciantes, é calcado principalmente nas aplicações de memória programável estática.

Memória programável estática

A figura 4.31 é um diagrama bloco de uma memória programável estática típica do tipo usado no computador PAZ. Existem cinco componentes básicos de uma memória programável: 1) linhas de entrada de endereço; 2) entrada de dados; 3) saída de dados; 4) seleção de chip e 5) uma linha de habilitação de escrita ou leitura. As linhas de entrada de endereço são conectadas à via de endereço do computador. No caso de uma memória programável de N por M bits, onde N é o número de palavras e M é o comprimento de cada palavra, devem existir linhas de endereço suficientes para endereçar todos os N bytes. Por exemplo, em uma memória programável de 1K são necessários 10 bits para endereçar todos os 1024 bytes dentro desta memória ($2^{10} = 1024$). O chip de memória programável estática que contém poucos bytes de dados, tal qual uma memória programável de 64 bytes, obviamente requererá poucas linhas de endereço. Para uma memória de 64 bytes, são necessários somente 6 bits de endereço.

Devido a memória programável estática ter a função de permitir o armazenamento e a restauração de dados, precauções devem ser tomadas com as entradas de dados e saídas de dados do componente. As linhas de entrada de dados e saídas de dados são designadas como funções separadas.

Durante a função de leitura, o dado armazenado dentro da célula endereçada é mostrado nas linhas de saída de dados. Durante a função de escrita, o dado posto nas linhas de entrada de dado deve ser armazenado no endereço designado pelo código das linhas de entrada de endereços. Não é necessário que a memória programável estática tenha linhas independentes de entrada e saída de dados.

Em muitos casos, esses dispositivos são configurados com saídas de três estados. Os dados de entrada e os dados de saída podem ser ligados juntos através de uma via de dados bidirecional, ou eles podem estar nas mesmas linhas e multiplexados no tempo. A figura 4.31 mostra um método de três estados da via de dados. Durante uma função de leitura, as linhas de entrada de dados são desabilitadas internamente do componente da memória. O conteúdo da célula de memória endereçada pelas linhas de entrada de endereço estará disponível na saída, abastecendo diretamente a via de dados bidirecional. Durante uma escrita, o oposto é verdadeiro. As linhas de saída de dados são colocadas no modo three-state (o qual pode ser considerado como um circuito aberto) e nenhuma corrente será sugada da via de dados bidirecional. O conteúdo da via de dados bidirecional é armazenado na célula de memória designada.

Todas essas multiplexações de funções são dependentes dos sinais de leitura/escrita (read/write) e de selação de chip (chip-select). Nenhuma operação pode ocorrer sem que o componente de memória tenha sido selecionado através do sinal de chip-select. Para selecionar um determinado banco como descrito anteriormente, é necessário uma lógica de decodificação que habilite esse banco através da linha de chip-select. Uma vez que um chip ou um banco de chips tenha sido selecionado, o computador determina se o dado deverá ser lido ou escrito nessas posições de memória. Em uma operação normal todas as memórias programáveis estáticas são deixadas no estado de leitura, e somente serão habilitadas durante um comando de escrita através de um nível 0 na habilitação de escrita (WRITE/ENABLE).

A figura 4.32 é um diagrama de tempo detalhado dos ciclos de leitura e escrita na memória. O write/enable é uma combinação do memory request e do write. O read/enable é uma combinação do memory request e do read. A decodificação desses sinais e do chip select foi discutida anteriormente. Em sua forma básica, o PAZ tem 8 linhas de chip select, cada uma endereçando um banco de 1K da memory.

A figura 4.33 ilustra um mapa de memória do computador PAZ básico como configurado inicialmente, o PAZ contém 3K bytes de memória. As posições de 0 a 3FF é uma EPROM de 1K. As posições de 400 até BFF são posições de memória programável estática. A EPROM de 1K está configurada para ficar nas posições de 0 a 3FF, dessa forma o PAZ pode ser facilmente inicializado ao ser ligado. A memória programável para as posições a partir de 400 é considerada como sendo a memória programável do usuário. É recomendado pelo menos 2K para uma operação satisfatória. O PAZ trabalhará com 1K, mas é recomendado 2K para expansão de periféricos básicos.

A figura 4.33 também mostra como a memória é ligada ao computador. Todos os três bancos de memória são ligados em paralelo entre as vias de endereço e de dados. Cada banco tem um chip select decodificado separadamente.

Quando a EPROM é habilitada e o $\overline{MCSO}$ está no nível lógico 0, o dado da EPROM é colocado nas linhas da via de dados. Os outros dois bancos da memória estão no modo three-state e não têm efeito sobre a via. Quando o computador acessa a memória programável, o chip select para aquele determinado banco de memória é colocado no nível lógico 0, e somente aquele banco de memória tem acesso à via de dados.

Enquanto todos os bancos de memória poderão ter aplicados sobre si o mesmo endereço, somente o banco selecionado estará no modo ativo. Para o computador escrever em um banco de memória o fluxo lógico é similar. Você notará que existem linhas de write/enable chegando em cada um dos bancos da memória programável estática de 1K, mas não à EPROM de 1K. Uma EPROM de 1K somente pode ser escrita com uma interface especial. Portanto, somente é ligado o write/enable para as memórias programáveis.

Se, por exemplo, o computador fosse escrever na posição 400, o chip select para o banco 1 e o write/enable para o banco 1 deveriam estar ambos no nível lógico ϕ para permitir que o dado presente na via de dados fosse armazenado na posição de 400. Esse tipo de configuração de memória programável é three-state e multiplexada no modo de leitura, os dados saem do chip de memória programável; no modo de escrita eles entram no chip, e quando não selecionado o chip está em 3 state.

A partir deste ponto, já discutimos o diagrama de bloco da memória programável estática. Para se fazer um computador operacional, é necessário configurar essa memória com componentes reais. Infelizmente, quando o PAZ foi projetado, um chip de memória programável de 1K por 8 bits era extremamente caro. Portanto, esses blocos de 1K foram projetados a partir de múltiplos componentes. Dois chips de memória programável relativamente baratos e populares são o Intel 2102A (apêndice C3) e o Intel 2114 (apêndice C4).

O 2102A é uma memória programável estática de 1K × 1. Para configurarmos uma memória de 1K × 8 necessitaremos de 8 × 2102 ligados em paralelo. Por comparação, para configurarmos um bloco de 1K × 8 com 2114s necessitaremos apenas de 2 chips. Isto porque o 2114 possui uma densidade interna maior do que o 2102. Como o objetivo de qualquer projeto de computador; para se montar em casa, é obter facilmente componentes de linha, os 2114s são os componentes de memória programável recomendado para o PAZ. Os 2102s também funcionarão, porém, não compensa utilizá-los devido à fiação necessária para implementá-los ficando, então, mais acessíveis os 2114s.

A figura 4.34 mostra como duas 2114s são interligadas para produzir um banco de memória programável de 1K × 8. Elas compartilham da mesma linha de chip-select. As linhas de entrada de dados são divididas de forma que 4 bits de dados são armazenados em cada chip. Como cada uma possui uma capacidade de endereço de 1024 bytes, as linhas de 10 bits de endereço são comumente compartilhadas. Para a construção do PAZ básico, dois circuitos do tipo ilustrado na figura 4.34 devem ser construídos. A memória total para computador básico é de 3K. Esta pode ser expandida até 8K sem nenhuma decodificação de endereço adicional. Não é absolutamente necessário ter 2K de memória programável se o usuário desejar apenas checar a operação do sistema. Pelo menos a EPROM deve ser montada como um banco de memória.

A EPROM de 1K contém um monitor o qual permite o funcionamento do PAZ. Este monitor possui vários pequenos programas que são chamados sub-rotinas. Quando o programa principal chama uma sub-rotina, o endereço de retorno é colocado na pilha de software localizada na memória programável. Ao término da sub-rotina o processador central tira da pilha este endereço e retorna ao programa principal. Normalmente não mais do que 64 bytes são necessários para a pilha. Entretanto, não é mais problemático montar duas 2114 para um banco de memória de 1K × 8 completo do que tentar montar uma memória de 64 bytes.

Um banco adicional de 1K, designado como banco 2 pode ser adicionado por arbítrio do usuário. Este banco é necessário se você planeja escrever programas que ocuparão mais do que 1K de memória incluindo a pilha. Como o computador está atualmente configurado, 1K parece adequado; entretanto para os programas adicionais descritos neste livro, é recomendado 2K. Isto é especialmente verdadeiro quando uma área de buffer é necessária para comunicação com periféricos externos. O esquemático para a configuração final de memória está mostrado na figura 4.35. Este pode ser somado ao circuito das figuras 4.17 e 4.26.

Diferentemente das outras seções do computador, a memória não pode ser testada, exceto sob controle de programa. Teoricamente, as linhas de endereço podem ser ativadas e o dado lido ou armazenado, mas isto não é fácil. Os testes de memória ocorrerão após a montagem da seção de entrada/saída. Basicamente, ela será verificada primeiro apenas com a EPROM, depois então com a adição da memória programável. Mencionei anteriormente que a EPROM e a memória programável operam relativamente independentes. Enquanto um programa pode ser armazenado em PROM, este necessitará da memória programável para sua devida execução.

Em um programa pequeno que carrega o acumulador, escreve em uma porta de saída, e de novo retorne para si mesmo, sem nenhuma chamada de sub-rotina; a memória programável não é necessária. Este programa pode estar localizado na EPROM. O procedimento exato para este teste será descrito no final da seção de E/S.

Figura 4.31 Diagrama em bloco de uma memória programável estática de N X M bits.

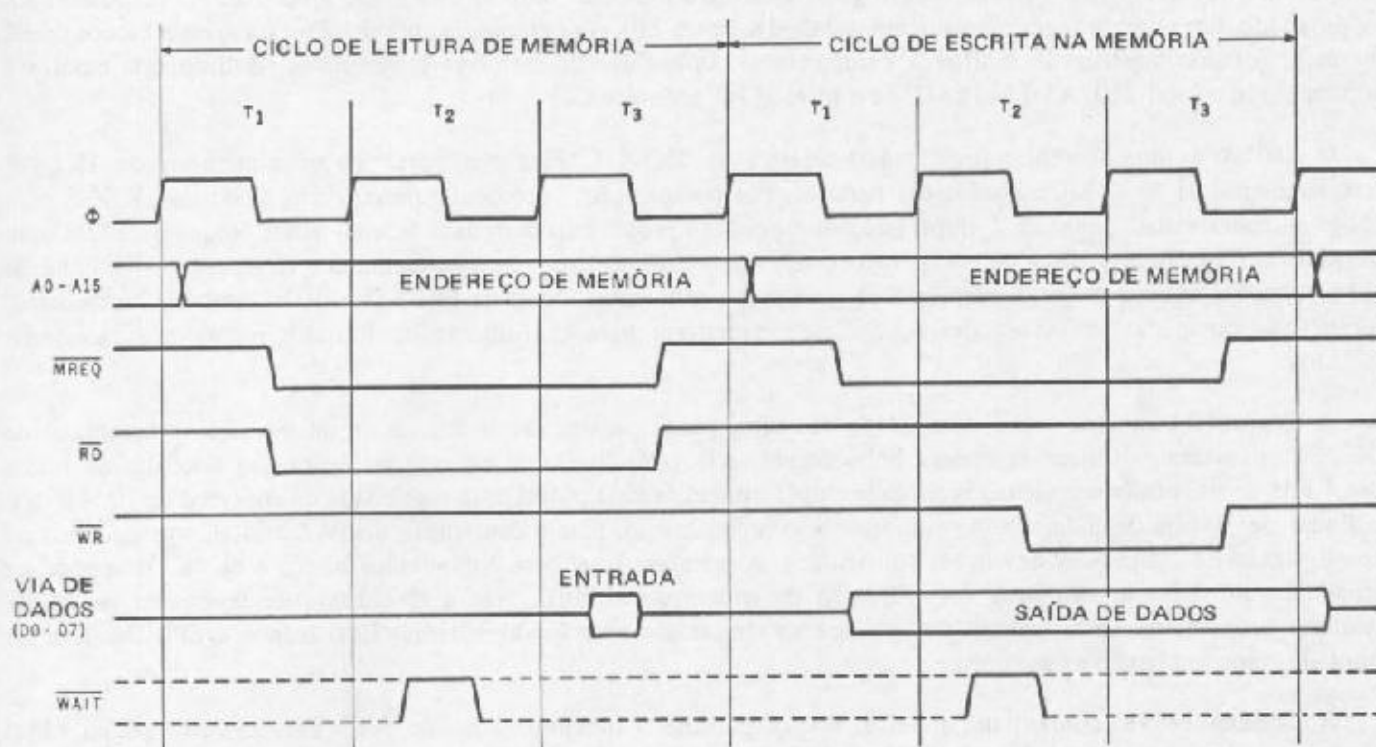



Figura 4.32 Diagrama de tempo dos ciclos de leitura ou escrita na memória para o Z80. Este diagrama não inclui os estados de $\overline{WAIT}$.

IV. Entrada/saída

Tanto quanto a discussão do controle do processador central e da decodificação de memória, são igualmente importante as funções de entrada e saída. Para o computador mostrar informações utilizáveis, este deve ser "interfaceado" com periféricos. "Interface" é um termo que se refere à capacidade de comunicação com dispositivos externos, tais como teclados, vídeo ou display de LED, e sistemas de memória de armazenamento. A comunicação pode ser feita através de dados de entrada ou saída.

As entradas de dados podem ser feitas através de teclados, memória de massa em cassete de audio, ou interfaces de aquisição de dados especiais. Similarmente, a saída de dados é feita do computador para os periféricos (displays de vídeo, leitoras numéricas, impressoras e interfaces de controle externo). A função e o formato da comunicação de dados entre o processador central e os periféricos podem variar consideravelmente, mas o caminho interno aos dados é fundamentalmente o mesmo.

Figura 4.33 Diagrama em bloco do mapa de memória para o computador PAZ.

Figura 4.34 Construção de um banco de memória programável de 1K X 8 utilizando-se 2 chips de memória programável 2114 de 1K X 4 bits.

Figura 4.35 Diagrama esquemático da configuração final de memória para computador PAZ básico.

O microprocessador Z80 possui tanto instrução de entrada quanto de saída. Uma saída do processador é logicamente o mesmo que uma escrita na memória, e a recepção de uma entrada originada em um dispositivo externo é similar a um comando de leitura de memória. Elas são diferenciadas das operações de memória pela combinação das linhas de *status* de leitura e escrita com a linha de controle de pedido de E/S. Uma concorrência lógica de um pedido de E/S e uma saída de *status* de leitura ou escrita determina a direção da comunicação com o dispositivo periférico. Simultaneamente com os sinais de controle, o código de endereço (1 entre 256) do periférico objeto é colocado na via de endereços. Um diagrama de tempos desses sinais está mostrado na figura 4.36. A lógica de decodificação foi detalhada na seção II deste capítulo.

Figura 4.36 Diagrama de tempo dos ciclos de entrada ou saída para o Z80.

O processo de montagem das portas de E/S no PAZ é constituído de dois estágios. Quando se monta um computador manualmente, a consideração mais importante é ver que a função de E/S funciona pelo método menos complicado. Um teste bem sucedido da seção de E/S também testa indiretamente a memória. Isto porque instruções de entrada e saída não podem ser exercitadas a não ser por um programa armazenado na memória.

As funções de entrada e saída do Z80 manuseiam 8 bits de cada vez, não importa se a configuração da interface externa é serial ou paralela. A transferência de dado entre o processador central e a E/S é em paralelo (8 bits) e basicamente ocorre como a seguir.

Instrução de saída

OUT(n), A

Quando esta instrução é executada, o conteúdo do acumulador A é colocado na via de dados e escrito no dispositivo *n*. O endereço do dispositivo *n* está localizado nas linhas de endereço de A_0 a A7.

Se o acumulador contém 40, em hexadecimal, quando a instrução OUT(23) A, for executada, 40 em hexadecimal será escrito no dispositivo periférico (também chamado "porta número") decodificado como 23 em hexadecimal.

Existem outras instruções de saída, mais complicadas, possíveis no conjunto de instruções do Z80; todas elas passam o dado através da via de dados para o dispositivo externo. Devido à via de dados ser usada para transferência de informação entre o processador central, memória e E/S, o computador deve ser liberado para continuar executando seu programa. O dado não pode permanecer na via de dados esperando pelo periférico (o processador central pode fazer isto, mas tais configurações seriam confusas no momento). O dado é válido somente por alguns ciclos de clock (relógio) e se for necessário um tempo maior, este deve ser armazenado.

O diagrama da figura 4.37 é um registro típico de armazenamento de 8 bits. Consiste de 8 elementos individuais de armazenamento com a entrada *store enable* comum a todos. Em sua forma mais simples uma célula de armazenamento pode ser um flip-flop tipo D tal qual mostrado na figura 4.38. A entrada de dado é ligada à linha de entrada D e é somente colocada para as linhas de saída (Q e $\overline{Q}$) durante um strobe de escrita de E/S. Utilizando-se 7474s necessitamos de 4 chips para uma palavra de 8 bits. Um método melhor é usar os circuitos da figura 4.39.

Instrução de entrada

IN A, (n)

Quando esta instrução é executada, o dado da porta (n) selecionada é colocado na via de dados e carregado no acumulador.

Se o dispositivo externo em questão possui o valor 10 em hexadecimal, quando a instrução IN A, 20 for executada, o valor 10 em hexadecimal do dispositivo número 20 em hexadecimal será carregado no acumulador.

Existem outras instruções de entrada mais complicadas, mas, como no caso das instruções de saída, o caminho para todos os dados é a via de dados. Para manter a via de dados dominada por um único dispositivo ligado a ela, é necessário que todos os dispositivos de entrada (isto é, a saída destes) sejam 3-state. Isto pode ser obtido utilizando interfaces lógicas, tais como UARTs e adaptadores para as interfaces de periféricos que sejam projetados para serem 3-state, ou pela adição de buffers 3-state de entrada tal qual ilustrado na figura 4.40 (diagrama de bloco típico de uma porta de entrada paralela de 8 bits).

O que estiver nas linhas de entrada de B_0 até B_7 durante uma instrução de leitura E/S será dirigido ao processador central. Usando esta instrução direta de leitura não existirá interação entre o processador central e o hardware externo ligado à porta de entrada. Uma lógica adicional é necessária para coordenar a temporização exata entre o computador e um periférico externo. A solução é chamada *handshaking*. Tal capacidade requer: que o hardware da porta de entrada seja mais sofisticado, conexão com o processador central, lógica de interrupção, ou portas de E/S adicionais para coordenar a temporização.

A aferição do hardware básico do PAZ é melhor efetuada utilizando-se o hardware menos complicado. Uma porta de entrada está ilustrada na figura 4.41 e consiste de 2 chips de 4 buffers 3-state. Poderá existir quem deseje ter um handshaking completo nas portas de E/S ou precise mais do que 8 mA de capacidade de saída de um dispositivo LS–TTL, podendo ser facilmente configurado utilizando-se o Intel 8212. As especificações descritas no apêndice C5 demonstram sua versatilidade.

Verificação da entrada/saída

Finalmente o PAZ pode ter um teclado, terminal CRT serial RS232, interface para cassete, e capacidade para E/S digital e analógica. Tentar ligar todos esses periféricos e testá-los simultaneamente está fora de cogitação. Uma forma mais metódica é construir o mínimo de hardware e software que o coloque operacional, adicionando, então, aos poucos o restante. Este é o caminho seguido.

Com exceção da memória, nós tentaremos eliminar qualquer problema potencial através de teste estático aonde for possível. Os dispositivos E/S das figuras 4.39 e 4.41 estão nesta situação. O teste completo de E/S necessita de uma porta de entrada e uma porta de saída. Isto pode ser montado como mostrado na figura 4.42. Somente a porta 0 precisa ser conectada no momento. O circuito adicional incluído neste diagrama pode ser ignorado. Levemos em conta apenas os CIs 21 até 23. Os outros dispositivos são melhorias para o PAZ básico e serão discutidos mais tarde.

Teste estático

Com a fonte desligada retire todos os CIs anteriormente instalados. Coloque os CIs 20, 21, 22 e 23. Ligue a fonte, coloque os sinais $\overline{DSOWR}$ e $\overline{DSORD}$ temporariamente em terra. Este artifício, impossível ao controle direto do computador, permite que a via de dados acesse simultaneamente às entradas e saídas da porta 0. Com a porta ligada

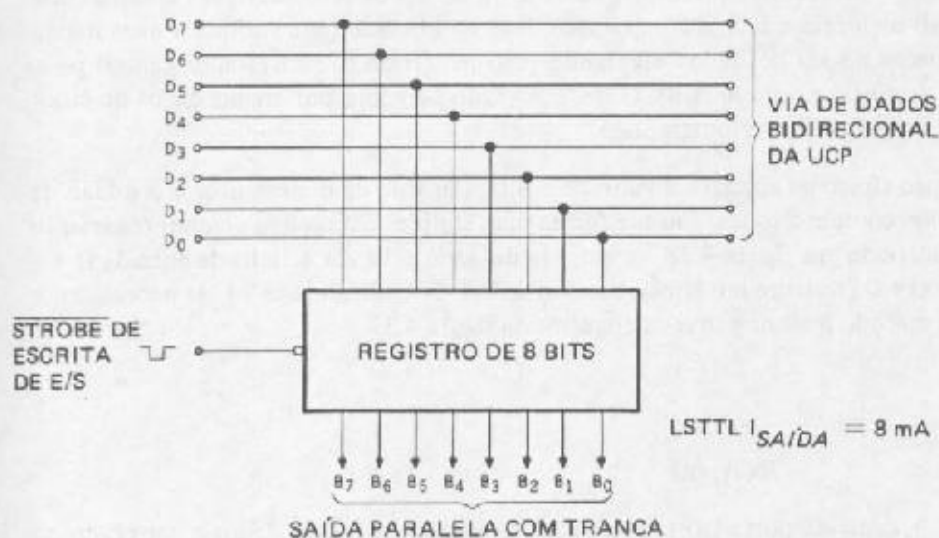



Figura 4.37 Diagrama em bloco de uma típica porta de saída paralela com tranca, configurada com um registro de armazenamento de 8 bits.

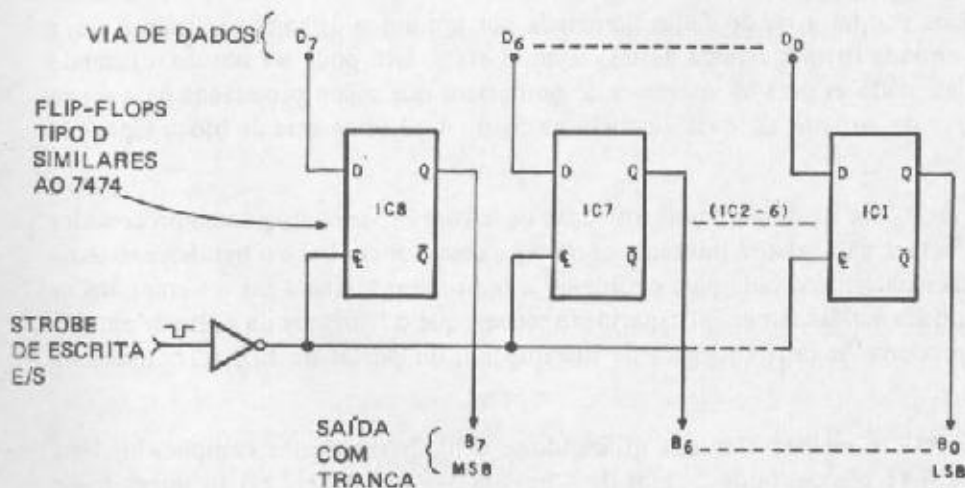



Figura 4.38 Diagrama em bloco de uma porta de saída paralela com tranca, utilizando flip-flop tipo D como um registro de armazenamento.

dessa maneira, o dado aplicado na entrada estará imediatamente na saída. Com as linhas de entrada dos CIs 21 e 22 abertas e a fonte aplicada, as saídas do CI 23 deverão estar em um nível alto. O aterramento seqüencial das linhas de entrada de B_0 até B_7 será refletido nas linhas de B_0 até B_7 do CI 23. Um teste final é desconectar o terra temporário em $\overline{DSOWR}$ enquanto uma das linhas de entrada do CI 21 e 22 é aterrada. O nível lógico 0 da saída do CI 23 deve permanecer baixo, mesmo quando a linha de entrada não estiver mais aterrada. Isto acontece porque o dado está "travado" e permanecerá assim, até que seja atualizado por outro strobe de escrita.

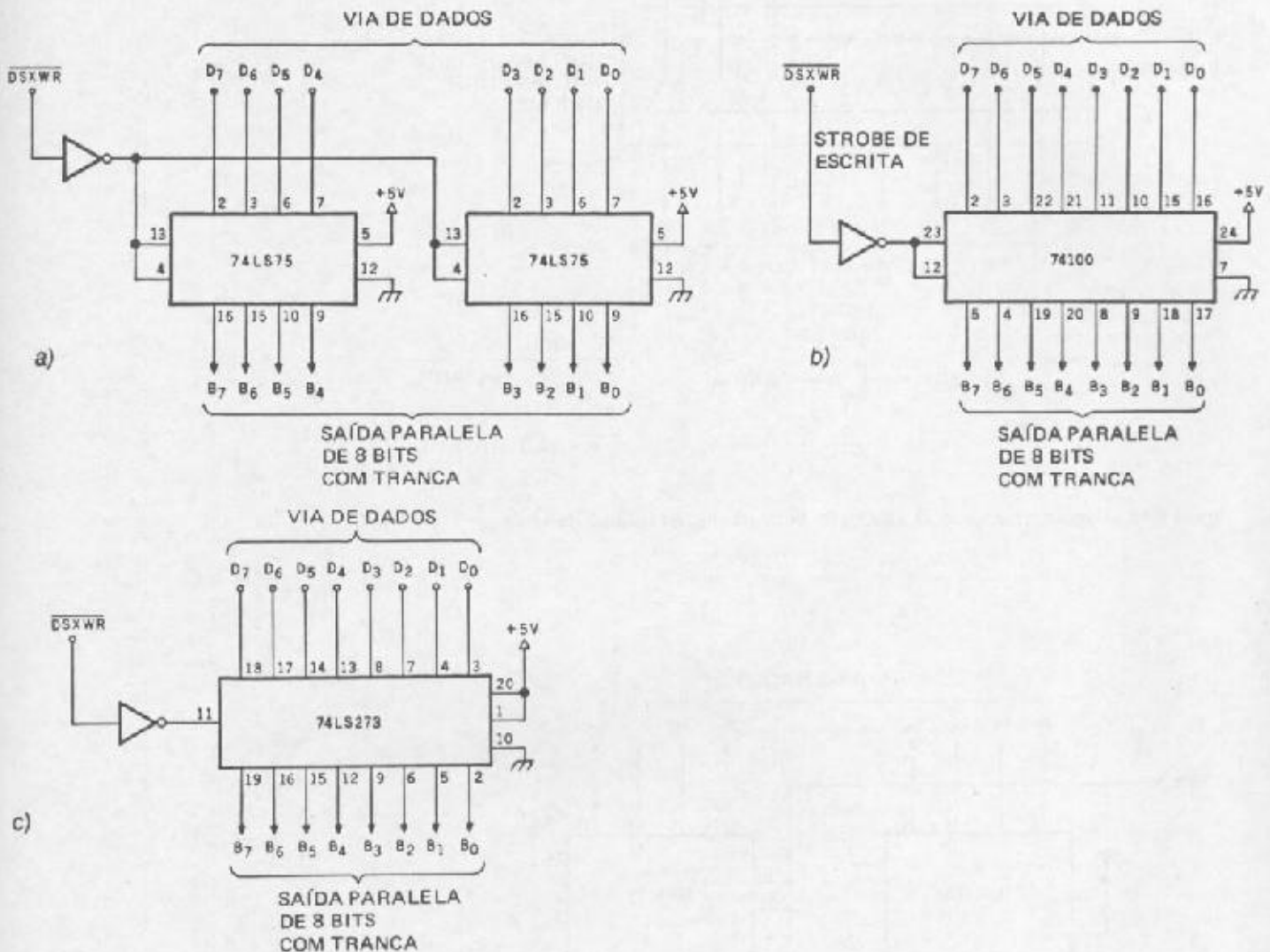



Figura 4.39 Diagramas esquemáticos de portas de saídas paralelas de 8 bits com trava.
a) Utilizando 2 chips LSTTL de 4 bits com tranca (LATCH).
b) Utilizando um LATCH TTL de 8 bits tradicional. Note que nenhum dispositivo LSTTL pode ser substituído, mas deve-se tomar cuidado quanto ao total da carga da via.
c) Utilizando um novo LATCH LSTTL de 8 bits.

V. Teste dinâmico do computador básico

Todo o sistema, com exceção da memória, deve ter passado com sucesso pelo teste estático. A montagem da memória deve ser testada por continuidade. Devido ao PAZ não ter painel frontal ou indicador (a menos que você deseje colocar um), o sistema completo só pode ser testado pela execução de um programa que exercite dinamicamente todo o hardware do sistema. Isto é mais fácil do que parece. Para o computador dar saída a um número para um determinado endereço de porta, o processador central deve estar operacional e pronto para executar a instrução. A leitura de memória deve funcionar ou o computador não saberá o que fazer. A decodificação da memória e E/S deve fazer com que o dado armazenado na memória chegue à porta de saída correta. E finalmente, para que o dado possa ser lido pela porta, a porta de saída deve estar funcionando bem, de sorte que, se você pode executar um programa, o computador está funcionando.

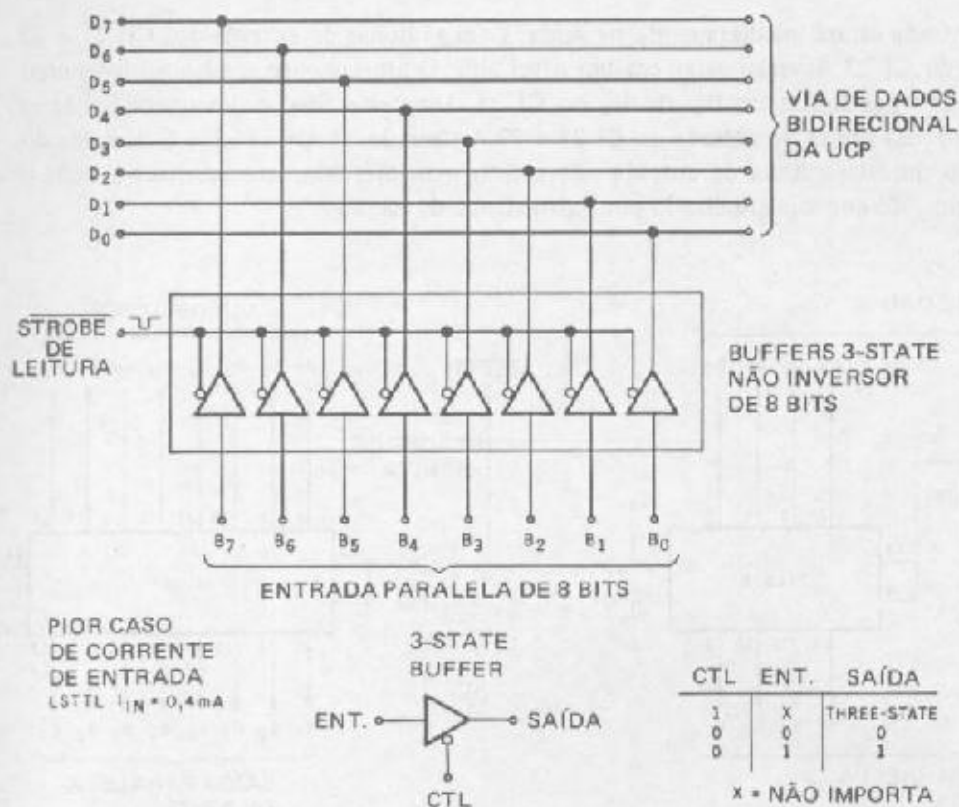



Figura 4.40 Diagrama em bloco de uma típica porta de entrada paralela de 8 bits.

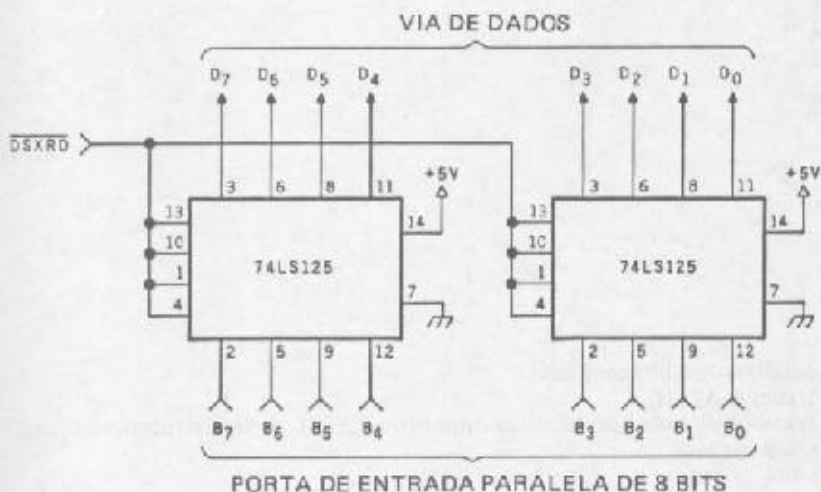



Figura 4.41 Diagrama esquemático de uma porta de entrada paralela de 8 bits para o computador PAZ.

Nós podemos ter um processo mais simples, utilizando um programa com o menor número de passos possível e por meio de eliminação inicialmente da memória programável. Lembre-se, o PAZ tem EPROM e memória programável. Sem monitor ou painel frontal, a memória programável não pode ser carregada diretamente a fim de se rodar um programa de teste. O programa de teste já deve estar carregado em ROM (no nosso caso em EPROM). Através de seleção cuidadosa das instruções usadas no programa de teste, a memória programável pode ser inteiramente deixada de lado quando nós rodarmos o primeiro teste. Para que complicar mais do que o necessário tendo-se mais hardware?

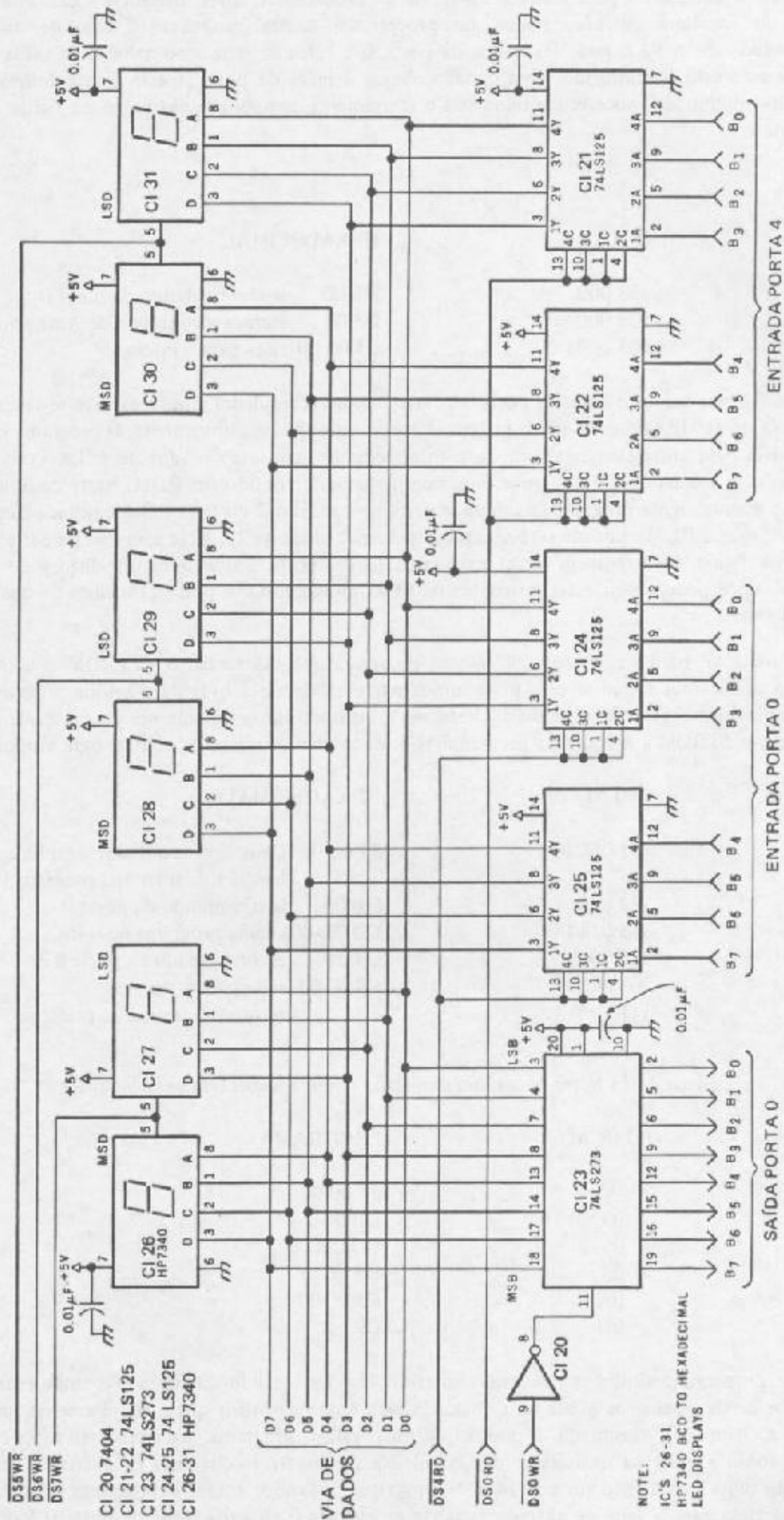



Figura 4.42 Diagrama esquemático de portas de entrada/saída para o computador PAZ básico com melhoramentos adicionais necessários para uso do software monitor do PAZ.

Poucas instruções são necessárias para testar a operação do processador; reset, memória e E/S. Normalmente o processador central ou funciona ou não. Falhas no processador central raramente é caso de uma instrução impropriamente executada. Se o PAZ pode ler dados da porta 0 e retornar o mesmo valor para saída da porta 0, podemos assumir que este está funcionando. Para o dado chegar à saída da porta 0, este deve caminhar através do processador central (assumindo que você tenha removido o aterramento temporário das linhas de strobe de E/S) sob o controle do programa.

O programa de teste é:

	OCTAL	HEXADECIMAL	
IN A, 0	333 000	DB 00	lê-se o conteúdo da porta 0
OUT 0, A	323 000	D3 00	escreve o conteúdo de A na porta 0
JP NN	303 000 000	C3 00 00	salta para o início

Este programa de 7 bytes vai ler o dado da porta 0 e carregá-lo no acumulador, então, escreve-se este mesmo dado na porta 0. A instrução de JUMP fará com que o programa repita esta ação continuamente. O programa não necessita de memória programável para armazenar nem um dado intermediário nem o apontador de pilha. Como somente o acumulador é afetado, o programa de 7 bytes pode estar completamente contido em ROM. Neste caso, a ROM pode ser uma EPROM 2708 manualmente programada como descrita no capítulo 7 ou uma ROM simulada como mostrado na figura 4.30. Se você usar a ROM simulada será necessário reduzir o clock de 2,5 MHz para compensar a capacitância do circuito externo. A figura 4.30 também inclui uma saída para a porta 5 que testa um display de dados a ser adicionado mais tarde. Você pode inserir estas instruções na EPROM agora, o que pode ser melhor do que reescrevê-la ou montar a pseudo-ROM.

O teste final para o PAZ básico é o exercício de um programa que utilize tanto a EPROM quanto a memória programável. De novo, a filosofia é que se este pode armazenar e recuperar 1 byte da memória programável, então todo este 1K deste banco deverá estar funcionando. Desta vez é utilizado um programa um pouco maior. O programa a seguir é armazenado em EPROM e a memória programável é usada pelo processador central para armazenar a pilha:

	OCTAL	HEXADECIMAL	
LD SP, nn	061 000 006	31 00 06	Coloca o apontador de pilha no meio do banco 1 da memória programável
IN A, 0	333 000	DB 00	lê o conteúdo da porta 0
CALL TEST	315 014 000	CD OD 00	chama programa de teste
OUT 0, A	323 000	D3 00	escreve o dado na porta 0
JP nn	303 000 000	C3 00 00	salta para o começo
TEST RET	311	C9	retorna ao programa principal

Quando montado, o programa de 14 bytes poderá ser carregado como a seguir (em hexadecimal):

POSIÇÃO	PROGRAMA
00/00	31 00 06
03	DB 00
05	CD OD 00
08	D3 00
0A	C3 00 00
0D	C9

A operação desse programa é similar ao exemplo anterior. Um byte é lido da porta 0 e então escrito de novo na porta 0. No interior dessas operações existe uma chamada para uma sub-rotina que é simplesmente uma instrução de retorno. Quando a chamada é executada, a posição de onde está o programa, para retomar a operação após a chamada, é colocada sobre a pilha na memória programável. Na conclusão da chamada (a instrução de retorno), o endereço é retirado da pilha e colocado no contador de programa podendo, então, o programa continuar de onde tinha parado. O único meio para o dado de entrada da porta de entrada 0 chegar à saída da porta 0 é pela execução apropriada desta chamada. Claro que isto necessita também do funcionamento da memória programável.

Muitos outros programas podem ser escritos para aumentar ainda mais os processos de testes. Pela minha experiência, entretanto, se estes dois programas forem executados, você pode contar com o funcionamento de tudo.

Uma vez atingindo este marco, você terá um computador operacional. O próximo passo é expandir esta unidade básica e tornar o PAZ mais versátil através da adição de displays de endereços e dados, de um teclado hexadecimal, de uma interface serial de acordo com uma operação de sistema que coordene as atividades destes periféricos. Enquanto o sistema atual pode ser montado em um cartão, para estas adições torna-se necessário um cartão para experimentos de projetos (Breadboard).

CAPÍTULO 5

OS PERIFÉRICOS BÁSICOS

Uma vez que o PAZ básico tenha sido construído e testado, estamos prontos para adicionar alguns periféricos necessários que irão aumentar grandemente a utilidade do sistema. Os periféricos externos facilitam a capacidade de entrada e saída do computador. Tais periféricos podem ser impressoras, tubos de raios catódicos (CRTs), fitas e discos. Entretanto, periféricos desta magnitude são normalmente usados em sistemas maiores. Para o nosso PAZ baseado no Z80, os periféricos usuais incluem um teclado para facilitar a entrada de dados e programas, um mostrador visual que permita ao computador indicar uma conclusão lógica em forma legível; uma interface de comunicação serial que permita ao PAZ comunicar-se com outro computador; e uma interface para audio-cassete que permita o armazenamento de massa. Estes quatro ingredientes fazem a diferença entre um cartão experimental e um computador pessoal utilizável.

O teclado pode ser tanto um bloco de chaves para entrada de dados limitada quanto um teclado alfa-numérico tipo "máquina de escrever" ASCII (American Standard Code for Information Interchange) para edição de textos e programação em linguagem de alto nível. O display usual pode ser desde um mostrador LED hexadecimal até um terminal CRT de 24 linhas por 80 caracteres. A porta serial, em conjunto com a interface para cassete, pode ser usada para inicialização do computador e para carregar programas aplicativos.

Da mesma forma que os circuitos anteriores, tentaremos colocar várias alternativas de projetos para que você possa construir um sistema pessoal. Cada um dos quatro dispositivos periféricos serão explicados em detalhe e exemplos de projetos serão dados. Tanto entrada hexadecimal de função limitada quanto teclado ASCII completo serão explicados. No caso do display visual, nós discutiremos o LED octal rudimentar e um mostrador hexadecimal para o PAZ. Para uma interação visual mais sofisticada, é necessário um terminal CRT. Devido a esta unidade ser muito mais complicada do que um teclado ou um display LED, um capítulo inteiro foi dedicado a ela. Minha premissa básica é começar com o essencial, prover o entendimento de suas aplicações, e então partir para o mais complexo.

A expansão do PAZ básico para um sistema de microcomputador interativo requer a adição de um programa de software para sincronizar e exercitar os novos periféricos.

Este software é chamado monitor e será discutido em um capítulo posterior.

1. TECLADOS

O único modo do Z80 poder se comunicar com um dispositivo externo é através da sua via de entrada/saída anteriormente descrita. (Existem outros métodos, tais como acesso direto à memória, mas eles serão ignorados aqui.) Quando o processador deseja sinalizar ao usuário a ocorrência de um evento, ele pode fazê-lo pela mudança do nível de saída de 1 bit da porta de saída paralela. Por exemplo, o final da execução de um programa pode ser designado pela mudança de um nível lógico 0 para um nível lógico no bit 7 da porta 0. Utilizando-se este conceito,8 elementos separados podem ser individualmente designados e controlados a partir dos 8 bits de uma porta do PAZ básico.

A entrada de informação é feita da mesma forma. Os números de 0 a 7 podem corresponder a 8 chaves nos 8 bits de entrada da porta 0. Isto é mostrado graficamente na figura 5.1. Quando a chave do bit 7 é acionada, aterrando a entrada, a transição de nível lógico pode significar para o computador uma entrada numérica de 7; muitas aplicações em microprocessadores necessitam apenas destes poucos bits de E/S. Um controlador de semáforo, por exemplo, com as luzes vermelha, amarela e verde, necessitaria apenas de 3 bits de saída.

O programa para controlar as luzes poderia ter sido escrito, montado, e programado em algum tipo de armazenamento não volátil. Entretanto, o PAZ deve interagir com um operador humano de tal forma que os programas possam ser desenvolvidos e testados. A maior diferença entre o controlador das luzes de tráfego e o PAZ deve ser os periféricos e não a capacidade do microprocessador.

Em nosso exemplo, podemos colocar 8 chaves em uma porta de entrada. Para entrarmos com a informação, teremos apenas de escrever um pequeno programa que leia o dado da porta 0 para o acumulador e então armazená-lo ou agir sobre ele.

O capítulo do software monitor mostrará estas manipulações, mas um problema deve ser resolvido primeiro, ou seja, a sincronização dos periféricos com o computador.

Como o computador pode saber quando o dado nas chaves é válido ou não? E, poderemos fazer um temporizador em software ou hardware que leia a porta a cada segundo? Pode você, por exemplo, tocar as chaves em um tempo estabelecido ou fazer o computador esperar?

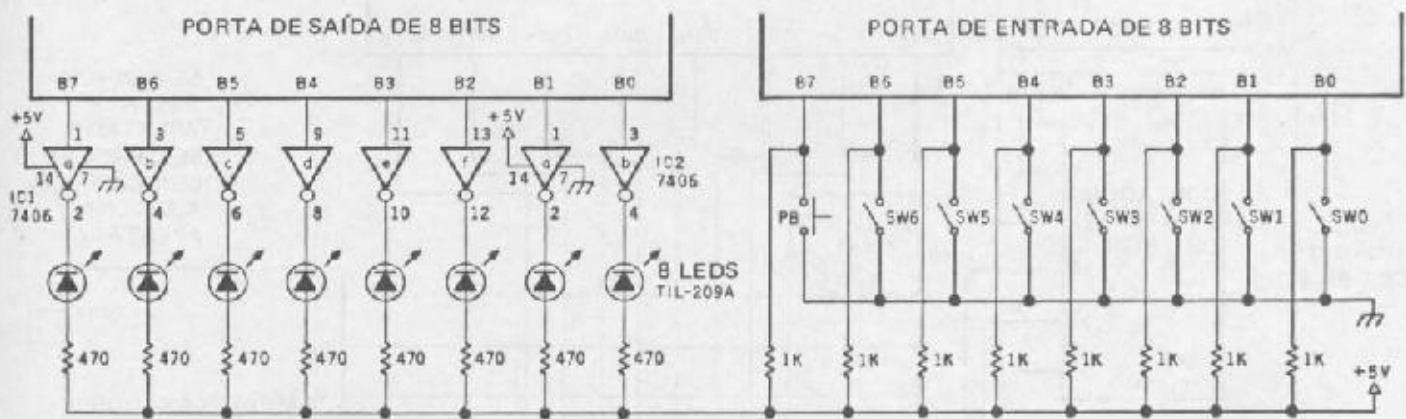



Figura 5.1 Uma interface de entrada/saída paralela com mostrador de LED e entrada chaveada.

O método mais popular de sincronizar um periférico que tenha entrada lenta de dados com um computador de rápida execução de programa é usar pulsos de carga de "dado pronto" (data ready). (Interrupções também podem ser usadas, mas elas envolvem uma programação complicada e não será aqui considerada.) O programa é escrito para ler e testar o nível lógico de somente 1 bit. Substituindo uma das 8 chaves, a do bit 7, por um botão, podemos simular a carga (strobe). Para fazer isto, primeiro coloque o dado nas outras 7 chaves; então, com o programa residente testando continuamente o bit 7, aperte o botão para gerar uma transição lógica. O programa, sentindo que o strobe do "dado ponto" está presente, lê o dado nos outros 7 bits.

Freqüentemente, não é prático limitarmos a sete interpretações simbólicas quando utilizamos 7 bits de entrada. O mais lógico é uma codificação da entrada e fazer com que os 7 bits representem 128 símbolos individuais. A escolha entre uma codificação versus uma entrada paralela direta é determinada pela aplicação. Se o computador é parte de um sistema de alarme, com cada bit de entrada representando uma chave de porta ou janela, então é importante saber as transições dos bits individualmente e simultaneamente. Para esta aplicação é necessário ter entrada de sinal paralela. Por outro lado, a entrada alfa-numérica de um teclado tipo máquina de escrever é por natureza serial, uma letra de cada vez. Portanto, não é aconselhável utilizar-se de uma entrada paralela de 128 bits para um teclado de 128 teclas. Uma codificação de 7 bits é mais eficiente.

O código para teclado mais utilizado é o ASCII (American Standard Code for Information Interchange). O apêndice B lista os códigos e os caracteres que estes representam. Qualquer teclado caseiro poderá refletir esta codificação, a fim de ser compatível com o software existente comercialmente como, por exemplo, BASIC.

Existem vários métodos que podem ser usados para gerar códigos de chaves compatíveis. As figuras 5.2 e 5.3 refletem uma aproximação do hardware e software, respectivamente. O diagrama de bloco descrito na figura 5.2 é um sistema de varredura hardware adequado para um teclado de 64 teclas. Um contador de 6 bits habilita progressivamente cada coluna enquanto varre todas as linhas em cada passo. Qualquer tecla apertada fará com que um nível lógico 0 passe por um multiplexador de 8 entradas até a lógica de controle de varredura. Este sinal é usado para gerar um strobe para o computador (também chamado de dado pronto). As linhas de endereço de coluna e linha do contador são lidas e indicam a matriz binária endereçada da tecla apertada. A compatibilidade com o código ASCII é simplesmente uma matéria de colocação da tecla certa no endereço correto dentro da matriz.

Um outro método de codificação adequado está descrito na figura 5.3. Esta técnica, que usa lógica de software para varrer a matriz, deve ser usada somente quando a velocidade de execução de programa do computador não é crítica. Enquanto reduz os circuitos para um único chip, passa a necessitar de uma porta de entrada e uma porta de saída. Esta funciona da mesma maneira apresentada na figura 5.2. O computador coloca no decodificador um código de 4 bits para o contador de coluna. Este, então, pesquisa a porta de entrada paralela para a linha com o nível lógico 0, significando uma tecla apertada. Enquanto isto parece ser uma forma fácil de decodificar 128 teclas, existem determinadas considerações de software.

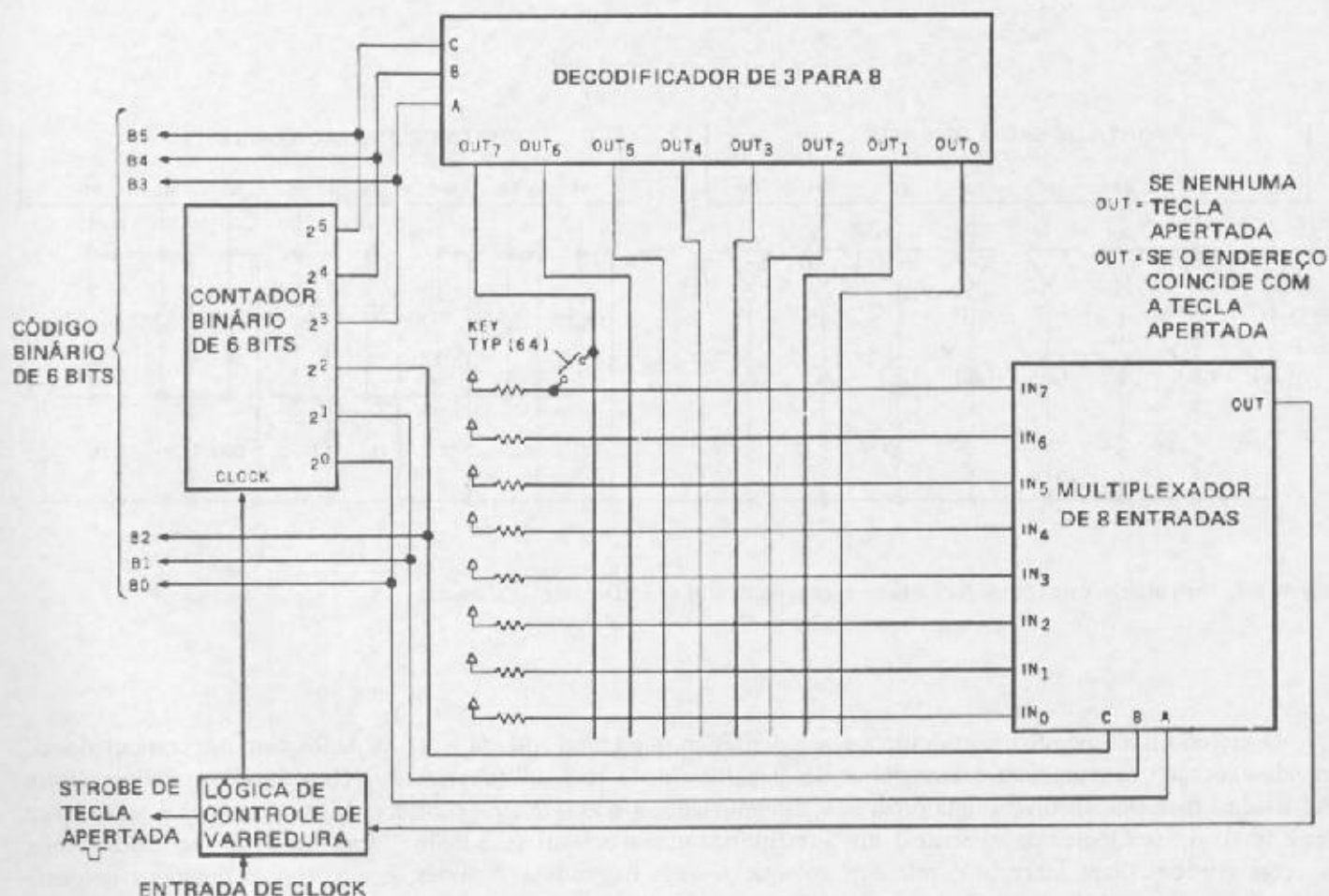



Figura 5.2 Matriz para varredura de um teclado de 64 teclas.

Figura 5.3 Circuito codificador para módulo de software de 128 teclas.

A tecla pressionada ou o strobe de dado pronto em qualquer teclado serve para dois propósitos, ou seja, significa que o dado está presente e pronto, e é temporizado de forma que o strobe não é gerado até que um período onde possa ocorrer ruídos tenha terminado. A razão para o retardo é óbvia. Lembre-se, esses microprocessadores podem executar 200.000 instruções por segundo. Um programa escrito para monitorar um strobe e ler o dado deverá passar umas cem vezes em uma única tecla por causa do ruído de contato. O contato mecânico poderá fazer com que apareça uns 100 strobes de dado pronto, se nós não tomarmos cuidado. Um strobe de dado pronto verdadeiro não é gerado até que tenha terminado um tempo onde não haja mais ruído e então será gerado um pulso com um rápido tempo de subida (<200 ms), com uma relação maior do que o ciclo de tempo do computador. A duração do pulso deve ser grande o suficiente para permitir ao programa captá-lo mesmo se este não estiver executando outra tarefa, e curto o suficiente para que o processador central não veja o mesmo strobe duas vezes.

Existem duas técnicas para combater o problema da duração do strobe. Uma é colocar um flip-flop gatilhado pelo strobe e ligar a linha de clear do flip-flop em um bit de saída. Depois de ler o dado, o programa pode retirar a condição de "dado pronto" através do reset do flip-flop. Isto normalmente é utilizado em casos onde o tempo de resposta para o teclado ou outro dispositivo é variado. Este método também garante que um evento será registrado e não perdido devido aos retardos de tempo. Claro, muitos codificadores de teclado não armazenam seus dados de saída. Se uma tecla não está apertada, mesmo se o strobe tiver sido colocado em um flip-flop, nenhum dado estará presente quando o computador for ler o teclado. Existem meios de se contornar isto, mas eles envolvem hardware adicional.

Normalmente o problema é ler o strobe duas vezes e não esperar o tempo suficiente para o reconhecimento dele. Ao invés de usar um flip-flop, muitos programadores utilizam um flag de software, a segunda técnica não se importa com a duração do strobe. Quando um strobe de tecla pressionada é sentido, o programa liga um flag em uma posição de memória, lê o dado, verifica então, de novo o strobe. Se o strobe está alto, o flag é verificado e o dado não é lido. Somente quando o strobe retorna ao nível lógico 0 é que o flag é desligado, permitindo uma nova entrada de dado.

Não é fácil construir um codificador para teclados ASCII de 64 ou 128 teclas. É mais fácil utilizar-se de um comercial tal qual o documentado no apêndice C6.

Muitas pessoas podem considerar o PAZ como uma ferramenta de aprendizado que pode ser eventualmente expandido para um sistema de microcomputador completo. Um teclado ASCII completo de 128 teclas pode vir a ser tão caro quanto o computador PAZ. Para minimizar o custo, sugerimos como o primeiro nível de expansão um teclado limitado, adequado para entrada hexadecimal. Com um número limitado de teclas para codificar, os circuitos TTL oferecem uma razoável vantagem de custo sobre os dispendiosos codificadores de memórias de leitura exclusiva.

A figura 5.4 é uma interface para teclado hexadecimal projetada especificamente para o monitor do PAZ. Um teclado hexadecimal permite a entrada de dados e instrução como números hexadecimais de 2 dígitos. Em conjunto com as 16 teclas numéricas existem 3 teclas de comandos designadas como *Exec* (para execute), *Next* e *Shift*. *Exec*

e *Next* serão explicadas na seção do monitor. O *Shift* é similar ao de um teclado normal e é usado para dobrar o número de códigos das teclas, permitindo, assim, um *Shift1*, *Shift2* etc. O significado particular de cada código será explicado mais tarde.

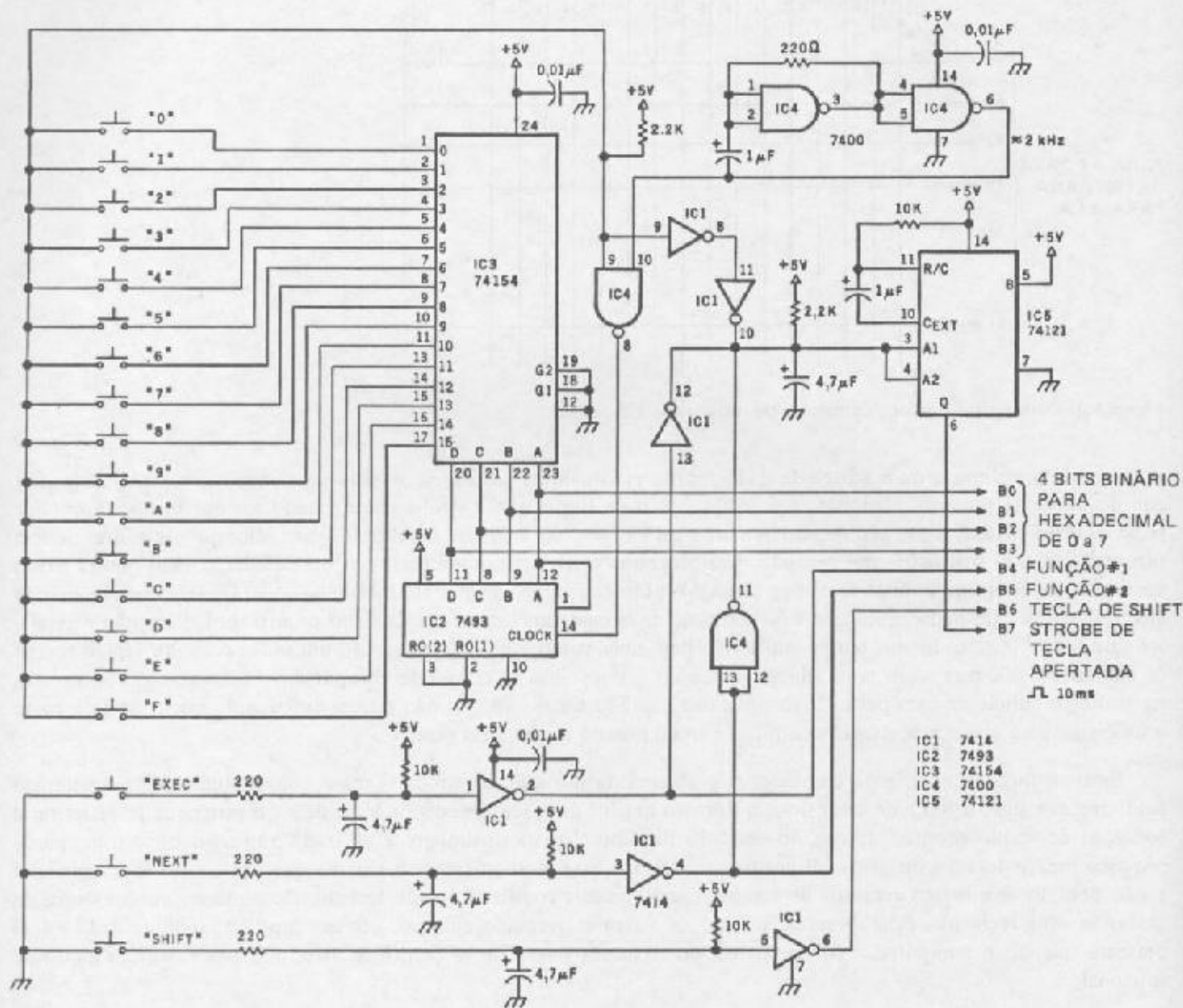



Figura 5.4 Interface para teclado hexadecimal.

O teclado necessário para suportar o monitor do PAZ possui 19 teclas. O codificador da figura 5.4 é uma combinação da varredura com a saída paralela. A codificação depende da tecla particularmente apertada.

As teclas hexadecimal de 0 a F são sentidas através do multiplexador IC2 e IC3. Através da contagem de IC2 é colocado seqüencialmente um nível lógico 0 em cada uma das 16 linhas de saída do IC3. Se qualquer tecla for pressionada, o nível baixo é enviado para o IC4 parando assim o clock. O contador é, então, travado no endereço da tecla que está sendo pressionada. A mesma ação que pára o clock também gatilha o IC5 que gera, assim, um strobe de tecla pressionada. As linhas de saída de B0 a B3 conterão o valor binário da tecla pressionada, enquanto o bit 7 é reservado para o strobe. As três teclas de função estão diretamente ligadas aos bits de entrada 4, 5 e 6. Três partes do CI1 servem para eliminar o ruído de contato. O EXEC e o NEXT são ligados de forma que gere um strobe de tecla apertada quando ativados. Por causa da tecla de SHIFT ser sempre usada em conjunto com outra tecla, esta não é conectada ao circuito de strobe.

É importante lembrar que a codificação desse circuito de 19 teclas não é ASCII. Um teclado ASCII não pode ser usado diretamente com o software monitor descrito neste livro, a menos que você use somente teclas ASCII correspondentes à codificação da figura 5.4, ou reescreva o software monitor para aceitar ASCII ao invés de códigos binários para cada tecla.

II. Adição do mostrador visual

Uma vez que um teclado tenha sido colocado no PAZ, estamos prontos para desenvolver um programa. O outro ingrediente chave é um mostrador visual que permita ao programador examinar as declarações de instrução e dado. A configuração menos dispendiosa é um mostrador de LED, preferencialmente hexadecimal devido ao software monitor ser escrito desta forma. Para os que preferem o octal também incluí um mostrador octal.

Mostradores hexadecimal podem parecer uma adição trivial para um sistema de computador dispendioso, mas algumas vezes estes dispositivos ajudam a tornar mais fácil a depuração do programa. Não pretendo que este substitua um CRT, porém é uma ferramenta necessária quando se depura um programa e uma necessidade para utilização do monitor do PAZ. Ele é de grande importância para mostrar rapidamente os dados do teclado ou de uma E/S com uma única instrução de saída.

Existem muitas maneiras de mostrar hexadecimal em um LED de 7 segmentos.

A figura 5.5 é um exemplo de um método grosseiro que utiliza uma PROM como um decodificador hexadecimal. (Se você desejar utilizar este circuito, um método de programação da 82S23 foi descrito no artigo "Um programador versátil para Memória de Leitura Exclusiva" na revista BYTE de novembro de 1975.)

Entretanto, este método utiliza um número excessivo de componentes e muitas pessoas podem não querer programar uma PROM. Uma alternativa é permitir que o computador faça a decodificação e alimente o display de 7 segmentos através de transistores ligados diretamente no latch de 8 bits da porta de saída. Uma outra forma necessitará da adição de circuitos lógicos extras em torno do decodificador de 7 segmentos. A primeira necessita apenas de um programa, enquanto a outra poderá envolver tantos componentes quanto os da figura 5.5.

Felizmente, existe um produto no mercado que pode resolver o problema. É o display LED hexadecimal HP7340 (da Hewlett Packard; existem outros displays equivalentes de outros fabricantes). Estes dígitos hexadecimais são feitos a partir do formato básico de 7 segmentos, utilizando-se pontos em vez de barras e sendo capaz de formar "B" e "D" em hexadecimal. Isto é possível pelo controle dos pontos dos cantos, que dá uma aparência arredondada. Esta habilidade discrimina um "B" de um "8" ou um "D" de um "0". Existem 16 caracteres diferentes e distintos.

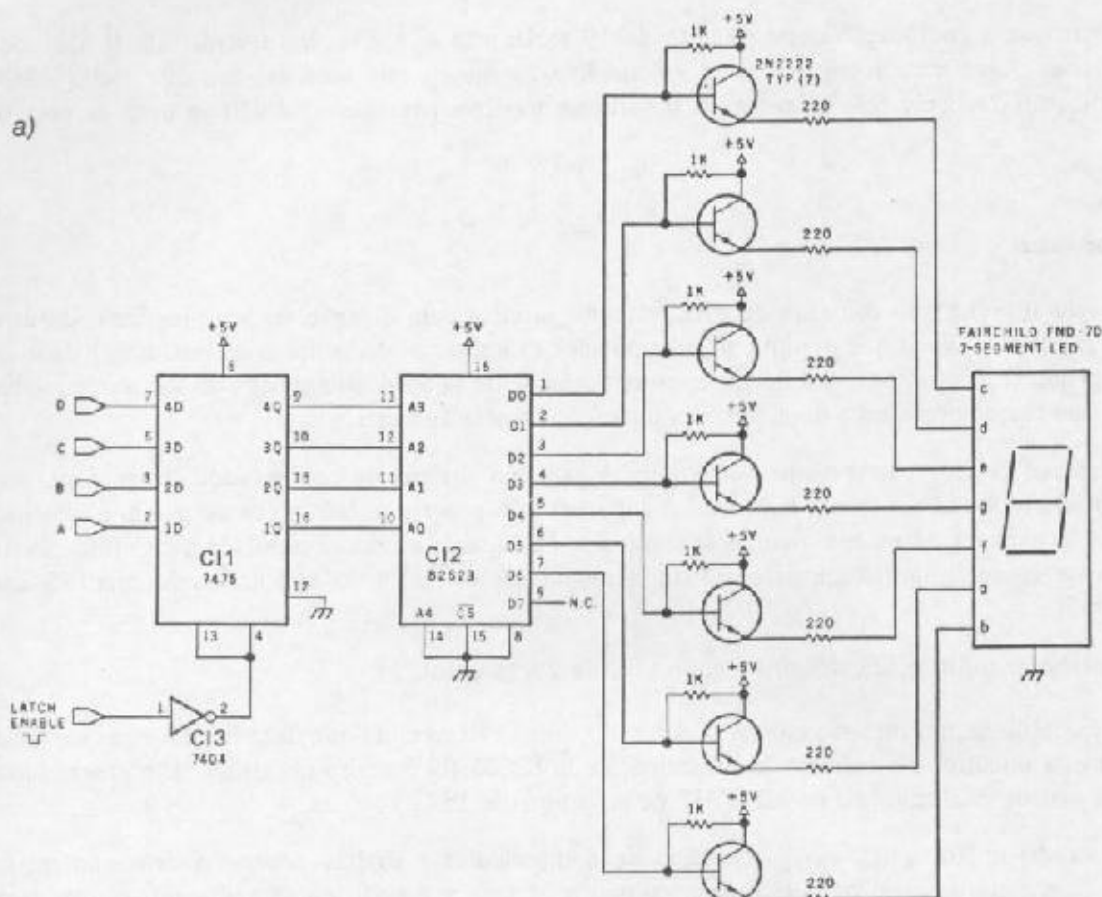
Uma facilidade adicional do HP7340 é que cada circuito display possui um latch/decoder/driver de 4 bits. Isto permite que o display seja ligado diretamente na via de dados. O resultado é um display hexadecimal de 8 pinos que executa com sucesso a função de todo o circuito da figura 5.5. As especificações dos pinos individualmente é dada na figura 5.6.

As figuras 5.7 e 5.8 demonstram como o HP7340 pode ser configurado para funcionar como uma porta de saída hexadecimal de 2 dígitos ou uma porta octal de 3 dígitos. Não é necessário um latch de 8 bits porque o componente já possui um. Os HP7340s podem ser ligados à via de dados tão simplesmente como qualquer outra porta de saída paralela, e são carregados a partir de um decodificador de chip-select, descrito anteriormente na seção de decodificação de E/S.

Para utilização do monitor software são necessários 6 displays hexadecimais (separados em 3 displays de byte). Três bytes são necessários para mostrar um determinado endereço H e L (alto e baixo) e o dado contido nesta posição. Os 6 displays hexadecimais devem ter as seguintes decodificações de carga:

Porta de saída #	Sinal Lógico	Parâmetro mostrado	CI #
5	$\overline{DS5WR}$	Campo de endereço do MSD	30, 31
6	$\overline{DS6WR}$	Campo de endereço do LSD	28, 29
7	$\overline{DS7WR}$	Campo de dado	26, 27

MSD – Most Significant Digit (Dígito mais significativo)
LSD – Least Significant Digit (Dígito menos significativo)

a)

b)

CÓDIGO DE ENTRADA				PROGRAMA DA 82S23								DISPLAY DE 7 SEGMENTOS
D	C	B	A	D7	D6	D5	D4	D3	D2	D1	D0	
0	0	0	0	0	1	1	1	0	1	1	1	0
0	0	0	1	0	1	0	0	0	0	0	1	1
0	0	1	0	0	1	1	0	1	1	1	0	2
0	0	1	1	0	1	1	0	1	0	1	1	3
0	1	0	0	0	1	0	1	1	0	0	1	4
0	1	0	1	0	0	1	1	1	0	1	1	5
0	1	1	0	0	0	1	1	1	1	1	1	6
0	1	1	1	0	1	1	0	0	0	0	1	7
1	0	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	0	1	1	1	1	0	0	1	9
1	0	1	0	0	1	1	1	1	1	0	1	A
1	0	1	1	0	0	0	1	1	1	1	1	b
1	1	0	0	0	0	1	1	0	1	1	0	C
1	1	0	1	0	1	0	0	1	1	1	1	d
1	1	1	0	0	0	1	1	1	1	1	0	E
1	1	1	1	0	0	1	1	1	1	0	0	F

Figura 5.5 Um método possível para um decodificador hexadecimal utilizando um LED de 7 segmentos standard.
a) Este circuito pode ser necessário para substituir um HP 7340.
b) O programa para o 82S23 (CI2).

Uma descrição mais completa de cada função do display é dada dentro da seção do monitor, e um esquemático completo mostrando como os 6 displays são ligados na via de dados está ilustrado na figura 5.9.

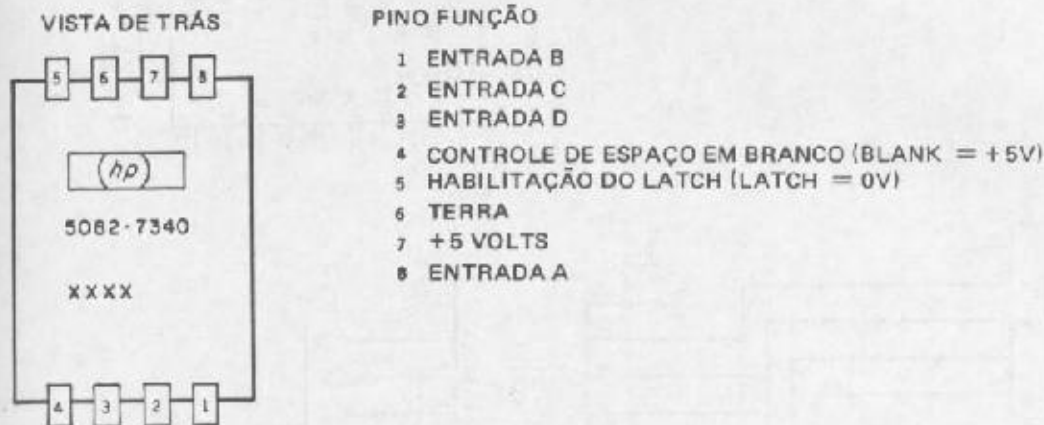



Figura 5.6 Posicionamento e funções dos pinos para o display HP7340. Displays similares são produzidos pela Dialite e Texas Instruments.

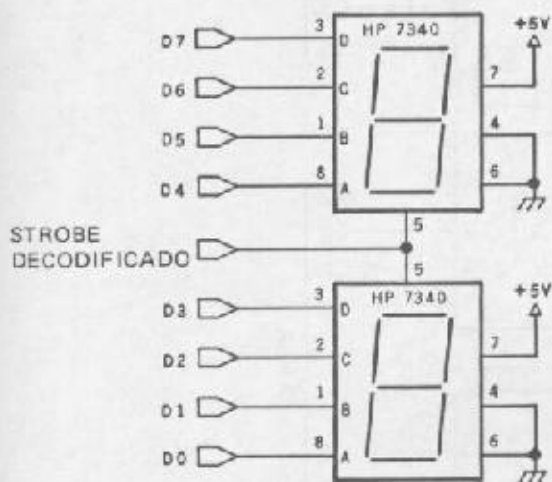



Figura 5.7 Display hexadecimal HP7340.

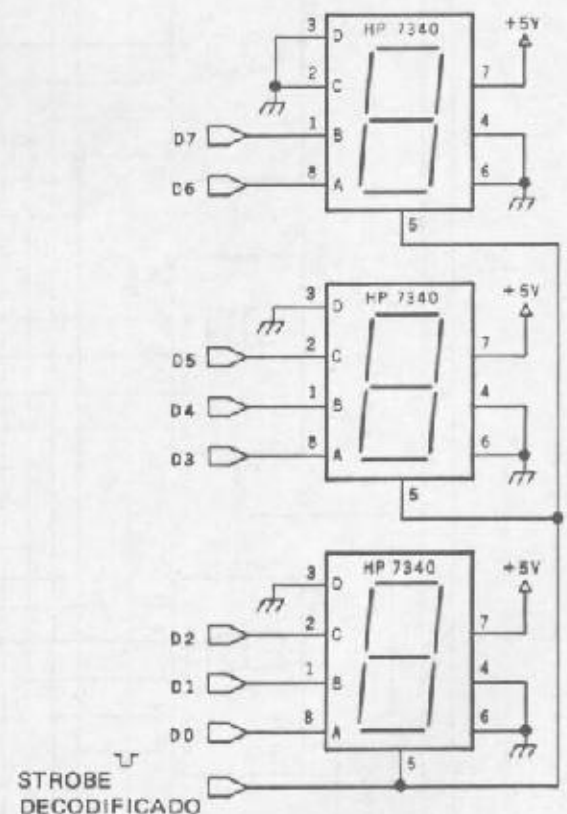



Figura 5.8 Display octal HP7340. O HP5082-7300 pode ser substituído pelo HP508-7340 em aplicações de display octal. O HP7300 mostra somente números.

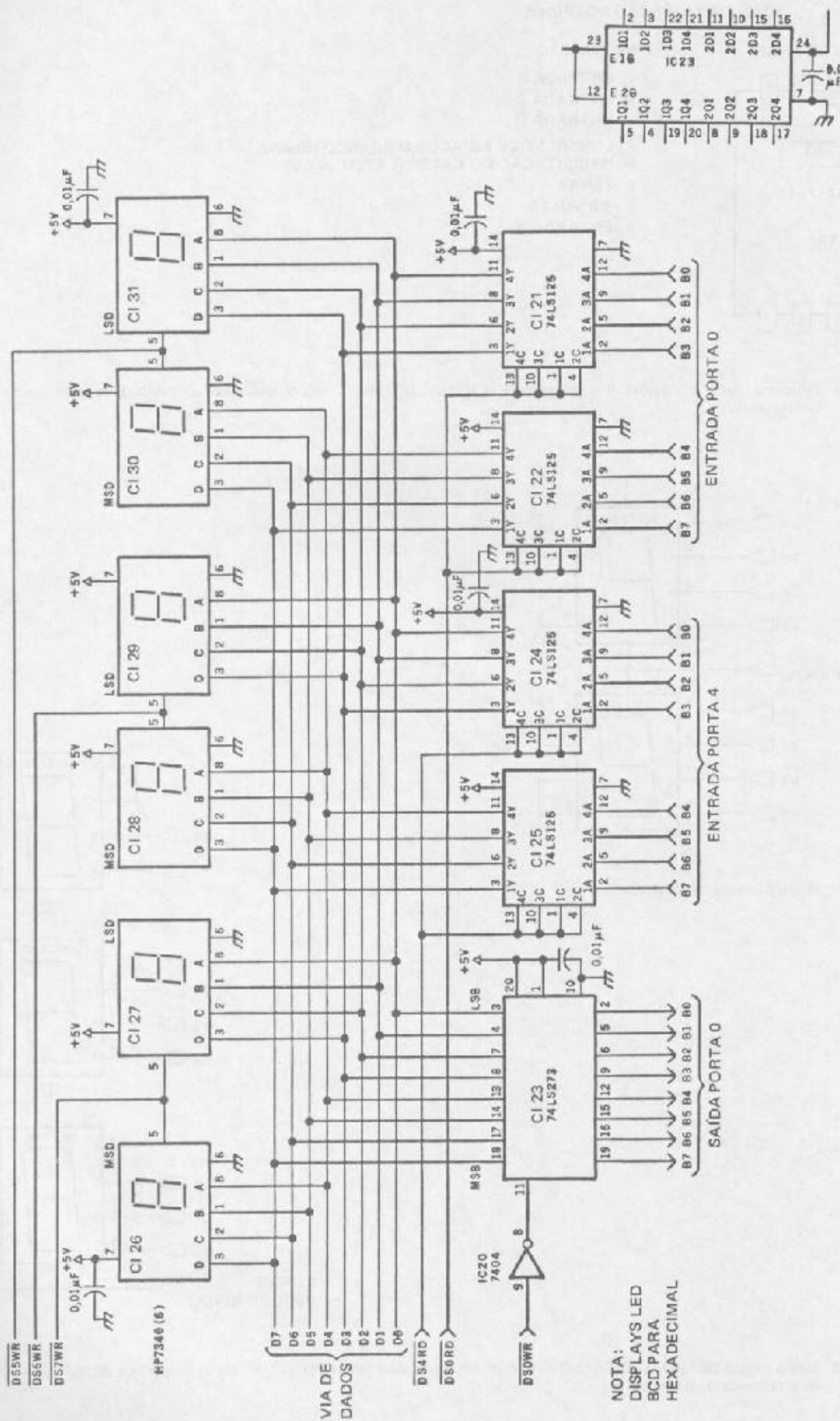



Figura 5.9 Diagrama esquemático de um display LED hexadecimal completo.

III. INTERFACE SERIAL

A capacidade de comunicação serial não é absolutamente necessária para fazer o PAZ funcionar, embora o monitor software descrito neste livro suporte uma interface serial.

Primeiro uma palavra sobre o conceito antes dos detalhes de projeto. Por que o PAZ necessitaria de comunicação? Quando nós discutirmos a interface serial para cassete, você entenderá que existem mais vantagens do que aparenta no momento. Se expansões futuras estão em mente ou se já se possui periféricos comerciais, tais como um CRT ou impressora, saiba que basicamente suas interfaces são do tipo serial.

Enquanto que a conversação com outros computadores através de linhas telefônicas necessita de um enlace serial, em geral periféricos como CRTs e impressoras também "conversam" serialmente. Portanto, projetando-se uma porta serial para colocar uma impressora, nós também obtemos a capacidade de conversar com outro computador.

Comunicação é simplesmente a transferência de informação de um dispositivo para outro. No caso de uma unidade CRT, o computador envia informação do caracter para a tela, enquanto o teclado permite ao usuário entrar com dados para o computador. No final de cada linha de comunicação deve haver um transmissor e um receptor. Em ambos os casos, a informação transferida é ASCII provavelmente consistindo de um código de 7 bits e, alguns casos, um bit de paridade para deteção de erro. Estes dados de 7 bits (ignorando o bit de paridade) aparecerão nas linhas de uma porta paralela. Estas 7 linhas mais uma referência de terra e um strobe (lembre-se que nós temos de avisar ao receptor quando o dado é válido) podem ser levadas até a entrada de um CRT. Mantendo-as como uma linha exclusiva do computador para o CRT, queremos agora uma linha similar entre a saída do teclado e uma porta paralela de 8 bits do computador; isto requer outras 9 linhas. Para complicar um pouco mais, vamos separar o terminal e o computador de aproximadamente de 300 a 400 pés, como deve acontecer em alguns sistemas comerciais. O resultado é que 400 pés de um cabo de 18 pernas (17 se você combinar as referências de terra) custará mais que o terminal. Lembre-se também que saídas paralelas TTL não devem ser usadas pra alimentar linhas maiores do que 20 pés sem buffers/drivers especiais; de outra forma poderá ocorrer erros de dados.

A solução para este problema é usar uma comunicação serial ao invés de paralela. O dado paralelo é convertido para serial, um bit de cada vez é enviado através de um par trançado de fios. Se para longas distâncias for necessário buffers/drivers, menos serão necessários se usarmos a comunicação serial. Bits de *start* e *stop* (início e fim), especialmente codificados, incluídos na transmissão serial, informam ao receptor que um dado válido está sendo enviado; para o exemplo acima, somente dois pares de fios para fazer a interação (tipo full-duplex); veja figura 5.10. No modo "half-duplex" isto pode ser reduzido a um único par trançado, mas a sincronização da linha de comunicação é mais complicada. Toda referência à transmissão serial que eu fizer estará limitada à operação full-duplex.

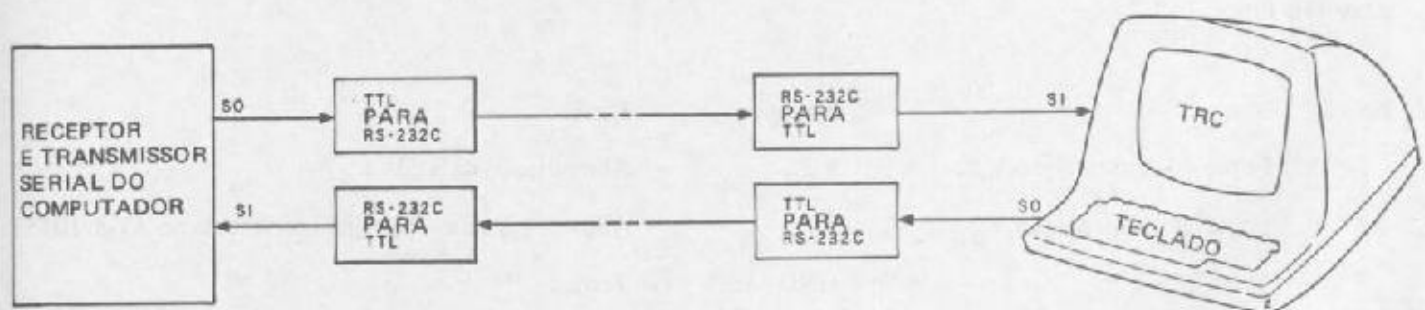



Figura 5.10 Diagrama de bloco de um enlace de comunicação RS-232C full-duplex.

Agora que já concordamos que a comunicação deve ser serial, como faremos a conversão de paralelo para serial? A resposta é um componente chamado UART (Universal Assynchronous Receiver/Transmitter). O Apêndice C7 fornece a especificação para o UART SMC COM2017 a qual é equivalente em função à AY-5-1013A (General Instruments). Para minimizar os requisitos de fonte de alimentação um AY-3-1015 ou TR1602 (Western Digital) de somente +5V pode ser substituído como eu tenho feito. A única mudança de especificação é que o pino #2 não é mais ligado a –12V.

A estrutura interna do UART consiste de um transmissor paralelo para serial e um receptor serial para paralelo separados, prendendo-se apenas por pinos de programação comuns. Isto significa que as duas seções do UART podem ser usadas independentemente.

A transmissão do computador para o TRC é feita de forma assíncrona e apenas em uma direção. O computador do mesmo modo recebe dados diretamente do teclado através de uma linha dedicada. Assim que o computador é ligado, este dispositivo de entrada comunica-se por dados paralelos, após uma reconversão para paralelo no UART.

A transmissão de dados atual segue o formato serial assíncrono ilustrado na figura 5.11. Usando o teclado como um exemplo, quando nenhum dado está sendo transmitido, a linha de dados é colocada em um ponto de espera (ou nível "1") de um strobe de tecla. O strobe de tecla apertada é um pulso positivo de 1 a 5 ms (este pode ser tão pequeno quanto 200 ns) indicando que uma tecla do teclado foi apertada, e que um código ASCII daquela tecla está pronto para transmissão. Este strobe de tecla apertada, o qual está ligado ao strobe de dados do UART, faz com que o dado ASCII seja carregado em um buffer de armazenamento paralelo e inicie o ciclo de transmissão do UART. A saída serial fará, então, uma transmissão de 1 para 0. Esta transição para 0 (start bit) permanece durante 1 período de clock e indica o começo de uma palavra transmitida serialmente. Após o start bit seguem os 8 bits de dados, cada um com um período de 1 clock. No final dos bits de dados, o UART envia os bits de paridade e stop para indicar o fim da transmissão. Se outra tecla é apertada, o processo se repete.

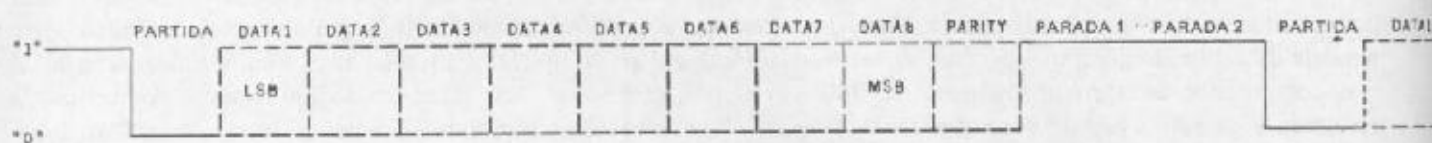



Figura 5.11 Um byte de dado como este é transmitido no formato serial assíncrono.

No término da recepção, o UART fica monitorando continuamente a linha de entrada serial até encontrar um bit de start. Isto ocorrendo, os 8 bits de dados são colocados em um registro e a paridade é verificada. Concluída a entrada serial, uma saída significando dado disponível é ativada pelo UART e pode ser usada como uma entrada de strobe para o computador. O UART não aceitará entradas seriais adicionais a menos que o flag de dado disponível seja reconhecido e a linha de reset de dado disponível seja ativada. A transmissão pode incluir ou excluir paridade, pode ter 1 ou 2 bits de stop, e o dado pode ser em palavras de 5 a 8 bits. Estas opções podem ser selecionadas através de pinos.

Pino #	Nome	Símbolo	Função
1	Fonte de Alimentação V_{CC}	V_{CC}	Alimentação de + 5V.
2	Fonte de Alimentação V_{GG}	V_{GG}	Alimentação de −12V (não conectada no AY-3-1015).
3	Terra	GND	Terra
4	$\overline{\text{Habilita Recebimento de Dados}}$	$\overline{\text{RDE}}$	Um nível lógico "0" na linha de habilitação do receptor coloca o dado recebido na linha de saída.
5 à 12	Dados recebidos	RD8 à RD1	Estas são as oito linhas de saída de dados. Os caracteres recebidos são justificados à direita. O bit menos significativo sempre aparece em RD1. Estas linhas têm saídas de três estados.
13	Erro de paridade	PE	Esta linha vai a 1 se a paridade do caracter recebido não corresponder com a paridade selecionada.
14	Erro de tamanho (Framing)	FE	Esta linha vai a 1 se o caracter recebido previamente não tiver bit de parada.

15	Sobreposição (Over-run)	OR	Esta linha vai a 1 se o caracter recebido previamente não tiver sido lido antes da chegada do nono caracter.
16	Palavras de *status*	$\overline{SWE}$	Um zero nesta linha coloca os bits de *status* (PE, FE, OR, DAV, TBMT) nas linhas de saída.
17	Clock de recepção	RCP	Esta linha deve ter como entrada um clock cuja freqüência seja 16 vezes a razão de recebimento dos caracteres.
18	Rearme dos dados	$\overline{RDAV}$	Um nível lógico baixo nesta linha limpa a linha de (DAV) dado disponível.
19	Dado disponível	DAV	Esta linha vai a 1 quando um caracter por completo houver sido recebido e transferido para o registrador de recepção.
20	Entrada serial	SI	Esta linha recebe os bits em série. Uma transição da MARCA (nível 1) para ESPAÇO (nível 0) é necessário para que se inicie a recepção dos dados.
21	Rearme externo	XR	Limpa os registradores de deslocamento. Liga SO, EOC e TMBT para o nível "1". Desliga DAV e os indicadores de erro para o nível "0". Limpa o buffer de entrada. Deve estar ligado ao nível lógico "0" quando não estiver em uso.
22	Buffer de transmissão vazio (TBMT)	TMBT	O indicador de buffer de transmissão vazio vai a 1 quando o registrador dos dados de transmissão está pronto para ser carregado com um novo caracter.
23	Pulso de dados (Data strobe)	$\overline{DS}$	Um pulso nesta linha faz com que os dados presentes nas linhas de dados sejam guardados no registrador interno. O início da transmissão destes dados é feito pela subida de $\overline{DS}$.
24	Fim de caracter	EOC	Esta linha vai a 1 a cada vez que um caracter é enviado completamente. Ela permanece neste nível até o próximo caracter começar a ser transmitido.
25	Saída serial	SO	Esta linha irá transmitir os bits serialmente. Permanece no nível 1 quando o caracter não está sendo transmitido.
26 a 33	Entrada de dados	BD1 a BD8	Existem oito linhas para entrada de dados.
34	Pulso de controle	CS	Um nível lógico 1 nesta linha faz com que os bits de controle sejam armazenados (EPS, NB1, NB2, TSB, NP). Esta linha pode ser pulsada ou ligada diretamente ao nível lógico 1.
35	Não há paridade	NP	Um nível lógico 1 nesta linha elimina o bit de paridade do caracter transmitido e do recebido. O bit de parada segue imediatamente o último bit de dados. Se não for usada esta linha deve ser ligado ao nível zero.
36	Número de bits de parada	TSB	Esta linha seleciona o número de bits de parada, um ou dois, que serão incorporados ao caracter quando da sua transmissão. Um nível lógico zero insere 2 bits de parada enquanto um nível lógico 1 insere um bit de parada.
37	Número de bits	NB2	Estas duas linhas serão decodificadas internamente

38 por caracter NB1 — para selecionar os seguintes números de bits por caracter.

NB2	NB1	BITS/CARACTER
0	0	5
0	1	6
1	0	7
1	1	8

39 Seleção de paridade par/ímpar EPS — O nível lógico neste pino seleciona o tipo de paridade a ser usado tanto na transmissão como na recepção. Um nível lógico "0" insere paridade ímpar e um nível lógico "1" insere paridade par.

40 Clock de transmissão TCP — Esta linha deve ter um clock cuja freqüência seja 16 vezes a freqüência que se deseja transmitir os dados.

A configuração final da interface é mostrada na figura 5.12. Como o UART é um componente de três estados, ele pode ser ligado diretamente à barra de dados. Os dados são escritos ou lidos como qualquer porta de E/S. Para o computador o UART se apresenta como uma porta de saída e duas de entrada que são: dados transmitidos, dados recebidos e *status*. Como todas as manipulações de dados, as transferências de dados são sincronizadas através dos pulsos de decodificação. Abaixo mostramos os endereços usados no PAZ para o UART

Porta #	Linha Lógica	Sinal
02 ENTRADA	$\overline{DS2RD}$	LÊ DADOS
03 ENTRADA	$\overline{DS3RD}$	LÊ *STATUS*
02 ENTRADA	$\overline{DS2WR}$	ESCREVE DADOS

Veremos em primeiro lugar a parte de hardware da interface serial. Quando o UART está ligado da maneira apresentada, não existe outra maneira de se operá-lo a não ser por software. Existem duas considerações a serem vistas: velocidade de transmissão e nível do sinal de transmissão. A velocidade de transmissão é comumente chamada bits por segundo. Tenha sempre em mente que são enviados geralmente 11 bits, oito de dados, 1 de paridade e 2 de parada. Existem já padronizadas algumas freqüências de transmissão que são mostradas a seguir.

110 bps
150 bps
300 bps
600 bps
1200 bps
2400 bps
4800 bps
9600 bps

Usando-se um CI gerador de freqüência e uma chave selecionadora mostrado na figura 5.12, o PAZ pode funcionar em qualquer uma destas freqüências. Em operação normal a maioria dos teletipos funcionam a 110 bps, impressoras como DECwriter II funcionam a 330 bps, MODEMs acústicos para telefone a 300 e terminais de vídeo de 1200 à 19200 bps. Como você pode ver, em teoria podemos nos comunicar com eles.

A razão de transmissão é apenas uma parte dos pré-requisitos da comunicação. Um computador pode ser feito todo usando-se nível TTL, mas uma interface pode ter nível CMOS (15V). Por isso é necessário ter-se um padrão de tensão para comunicação. O mais amplamente aceito e geralmente usado é o padrão EIA RS 232-C.

Embora níveis TTL possam ser usados para comunicação, eles não são adequados para sinais que tenham de percorrer mais de 3 a 6 metros. O problema se apresenta pelo fato que apenas 2V separam a lógica "1" a "0" e não por velocidade ou capacidade de corrente. Com apenas 2V de imunidade a ruído, a comunicação estaria sujeita à interferência de motores e chaves.

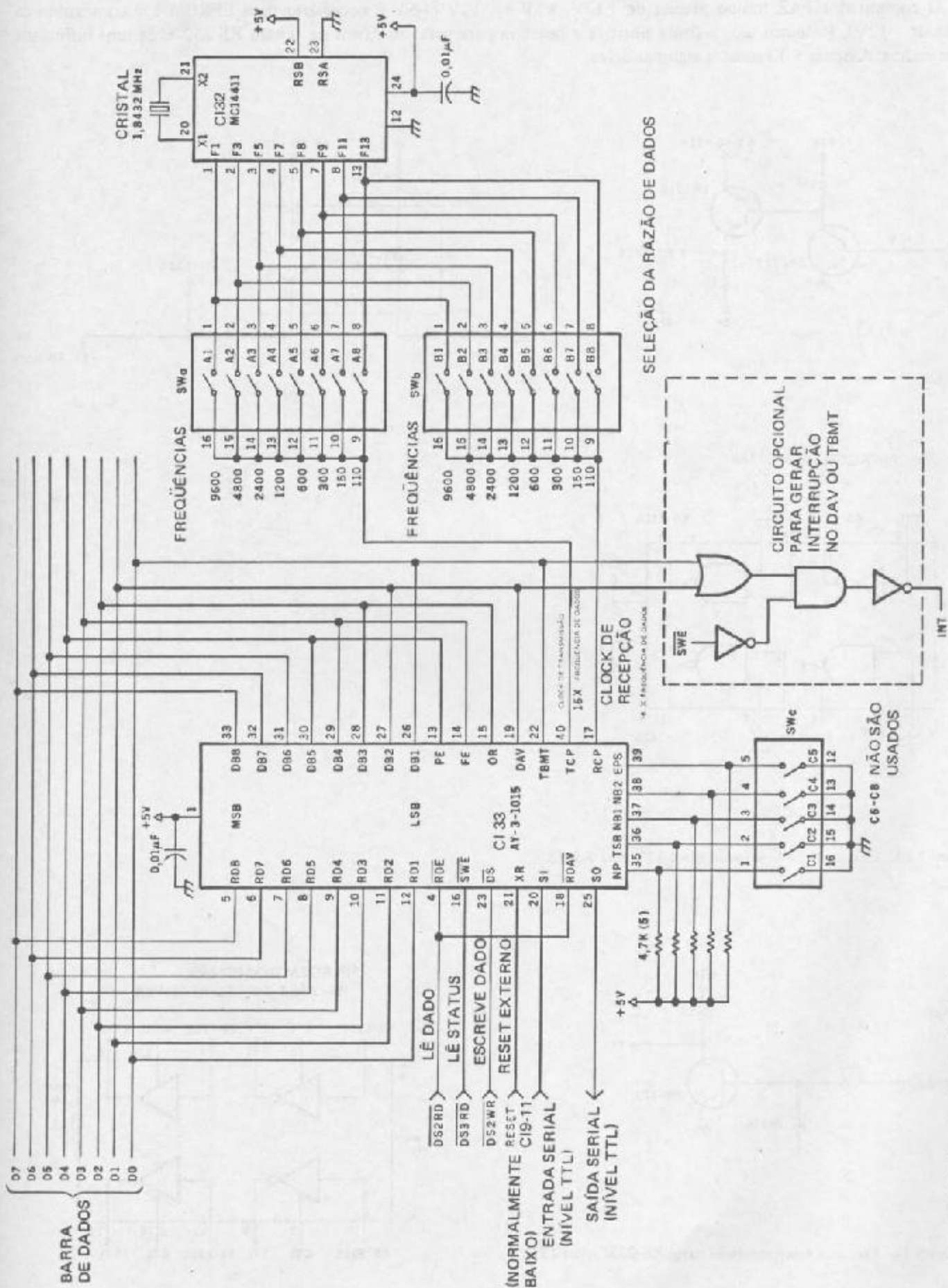



Figura 5.12 Configuração final da interface serial.

Uma comissão industrial estabeleceu um padrão para resolver este problema. Este padrão não se refere apenas aos níveis lógicos, mas também ao tipo de conector, impedância de carga etc.

Os níveis de sinal do RS 232-C vão de –3V a –15V para representar o nível lógico de 1 a +3V a +15V para representar o nível lógico zero. A região de –3V à +3V é a região de imunidade a ruído.

O computador PAZ básico precisa de +12V, +5V e –12V (–5V é necessário para EPROM e é conseguido da fonte de –12V). Podemos usar a fonte positiva e negativa para gerar os níveis de tensão RS 232-C de uma infinidade de maneiras. A figura 5.13 mostra algumas delas.

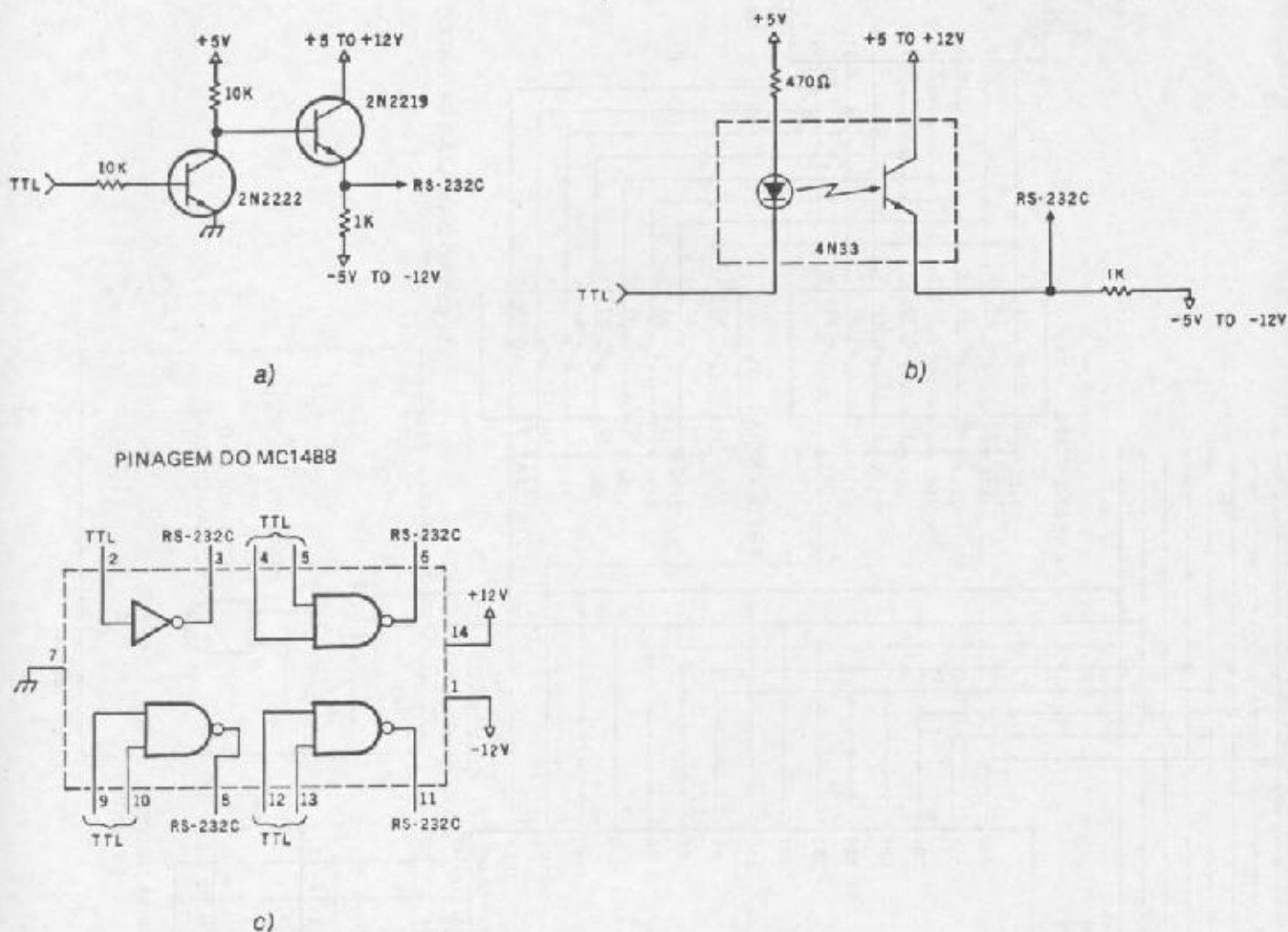



Figura 5.13 Circuitos conversores de nível TTL para RS-232C.

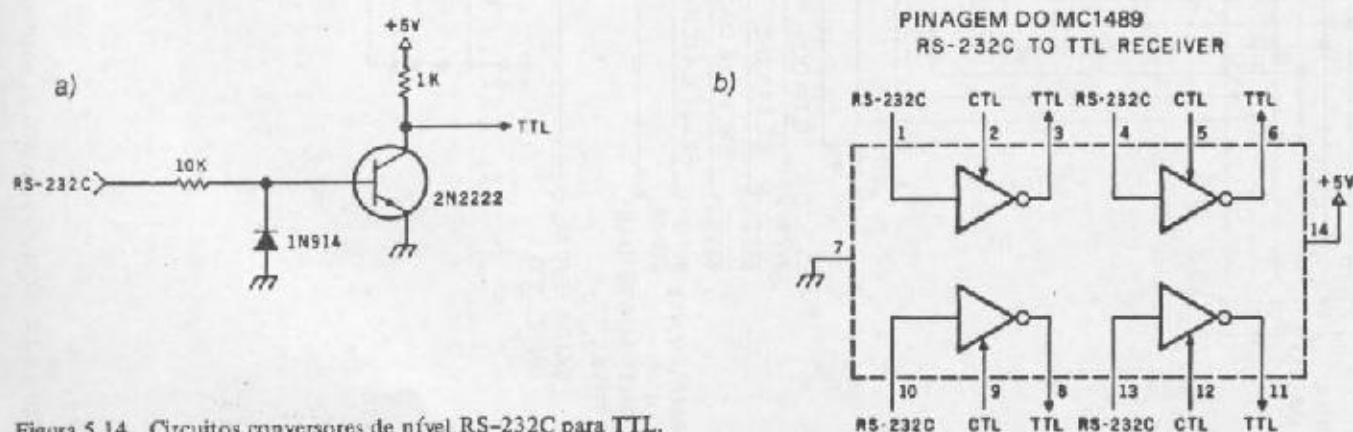



Figura 5.14 Circuitos conversores de nível RS-232C para TTL.

IV. INTERFACE PARA CASSETE

Com o teclado e o mostrador um operador será capaz de escrever alguns programas, mas, se eles não forem transferidos para um componente só de leitura, (por exemplo, memória do tipo ROM) estes dados serão perdidos quando o equipamento for desligado. Naturalmente o computador pode ser deixado ligado constantemente. Mas o que aconteceria se você quisesse desenvolver um outro programa que ocupasse a mesma posição de memória? A melhor solução seria ter um meio que armazenasse temporariamente uma grande quantidade de danos.

Em computadores grandes isto é conseguido através de discos e fitas magnéticas. Mas estes equipamentos são muito caros para o hobista, uma alternativa mais barata é o uso de cassetes de audio como sistema de armazenamento.

Em geral uma interface de armazenamento em cassete consiste basicamente de três partes: um transmissor/receptor serial; um circuito que converte o sinal TTL serial em um sinal compatível com o cassete de audio, e um programa de aplicação que escreva e leia do cassete. A configuração básica é mostrada na figura 5.15.

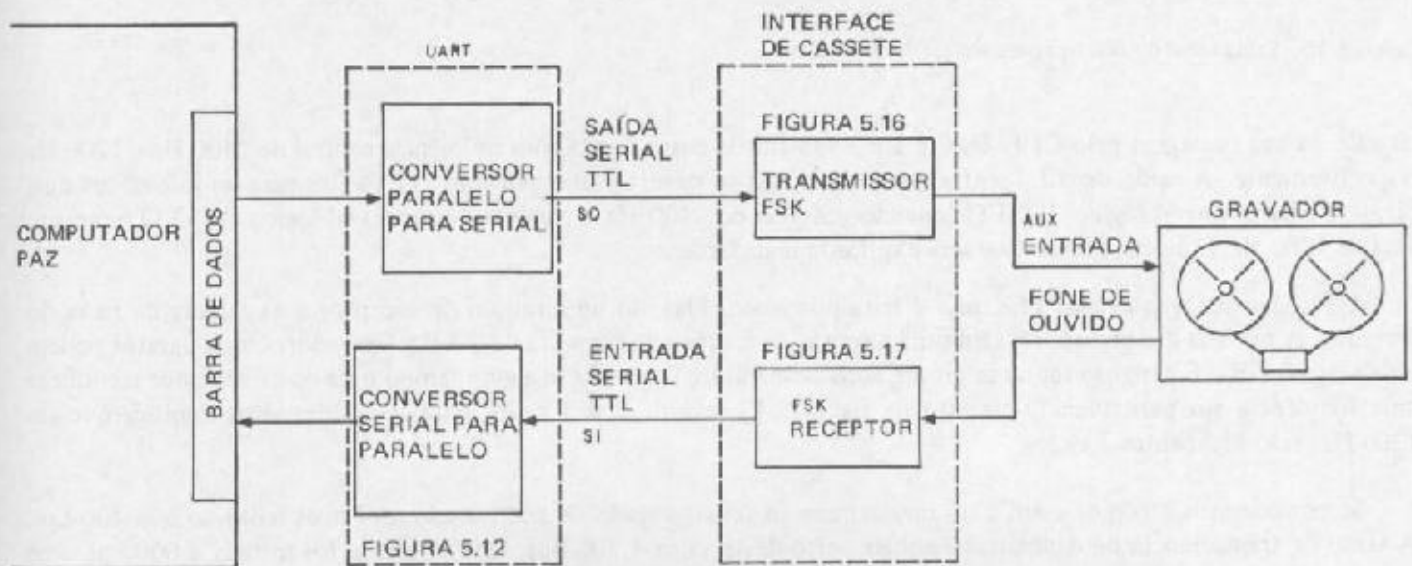



Figura 5.15 Diagrama em bloco de um sistema de armazenamento em cassete.

O transmissor/receptor serial é nada mais nada menos que um UART (transmissor receptor assíncrono Universal). Com os conversores MC1488 e 89 ligados às suas linhas seriais ele se comunica via RS-232C. Entretanto se você acopla uma interface para cassete a estas linhas, ele pode se duplicar como um elemento de armazenamento. Um benefício adicional é que os dados gerados serialmente pelo UART oferecerão alguma compatibilidade entre sistemas de computação pessoal.

A saída do UART é em nível TTL. Mesmo com os drivers RS-232C a saída lógica é ainda um nível CC. Como gravadores de audio não podem gravar CC, a saída do UART deve ser convertida de alguma forma. A solução é FSK (Frequency Shift Keying – Deslocamento de Freqüência). A saída TTL do UART é convertida em tons de audio. Uma freqüência representa o nível lógico 0 e uma outra o nível lógico 1.

A figura 5.16 mostra um circuito que produzirá uma freqüência. Uma freqüência de 4800 Hz é derivada do MC14411 que é o gerador de razão de dados já instalado. O CI 2A e 2B funcionam como divisores programáveis. Com um nível TTL de 1 na entrada do CI 2, este divide por 2 os 4800 Hz, dando uma saída de 2400 Hz. Quando o nível lógico é mudado para 0, ele divide por 4, produzindo uma saída de 1200 Hz. As freqüências FSK são geradas a uma razão da saída serial de 300 bits por segundo, e são conectadas diretamente ao gravador através do microfone ou da entrada auxiliar. Estas freqüências e a razão de dados são geralmente referidas como Padrão de Kansas City.

Para se obter de volta os tons que foram gravados na fita requer-se um circuito como o da figura 5.17. Em geral este circuito consiste de um par de filtros passa banda e um comparador de tensão. O gravador é colocado para um nível de saída de aproximadamente 1 VOLT pico a pico. Este nível não é crítico porque o sinal é amplificado e limitado

Figura 5.16 Saída serial de 300 bps para um gravador de audio.

quando da sua passagem pelo CI 1. Os CIs 2 e 3 são filtros passa banda com freqüência central de 2400 Hz e 1200 Hz respectivamente. A saída do CI 1 entra em ambos, mas só deverá passar por um. O CI 4 compara as saídas dos dois filtros e gera um nível lógico "1" TTL quando um Ton de 2400 Hz é recebido, e um nível lógico "0" TTL com um Ton de 1200 Hz. O ajuste da interfase será explicado mais tarde.

A escolha das freqüências FSK não é feita por acaso. Elas são uma função do receptor e da largura de faixa do gravador. A maioria dos gravadores têm uma resposta de freqüência em volta de 9 KHz. Gravadores mais baratos podem ter de 5 a 6 KHz. É perigoso tentar-se gravar Tons acima deste limite. Leva algum tempo para que o receptor reconheça uma freqüência em particular. O circuito da figura 5.17 leva de 2 a 3 ciclos para responder. Isto significa que em 1200 Hz serão necessários 3 ciclos.

Se considerarmos o pior caso, o de mandarmos só zeros, a razão de transmissão será mais lenta do que 400 bps. A razão de transferência de dados padrão mais perto deste valor é 300 bps. Esta interface foi testada a 600 bps, mas necessita-se de um alinhamento muito preciso para poder alcançar velocidades maiores.

Um ponto final a considerar é o software que irá exercitar o hardware. O monitor do PAZ não suporta uma interface de cassete diretamente. Até que você escreva o programa do cassete em uma EPROM, você terá de entrar na mão com um pequeno programa de carga.

Para ler o K7 a lógica do programa segue o fluxo mostrado na figura 5.18.

Primeiro um ponteiro é colocado nos registros H, L para designar onde os dados lidos do K7 serão armazenados.

Em seguida, utilizando-se a rotina de comunicação serial do monitor do PAZ, nós simplesmente chamamos "SERIAL IN" a qual retorna com um byte do UART. Este byte é, então, guardado na memória e HL é decrementado e comparado com um endereço de parada já determinado. Se não for igual, todo o processo será repetido.

Para escrever no K7 o processo é o mesmo e o fluxo é mostrado na figura 5.19. Outra vez um ponteiro é colocado no início e no fim da área de memória que se quer salvar, logo a rotina de "SERIAL OUT" é chamada monitor, a qual envia para o K7 o byte desejado. Finalmente o ponteiro é decrementado e comparado com o endereço final para ver se chegou ao fim da transmissão.

Essas rotinas são relativamente fáceis de se escrever e poderão ser colocadas nos endereços vazios da EPROM do monitor. Seja qual for o caso você sentirá a versatilidade e a capacidade que uma simples interface de K7 adiciona a seu sistema.

SINTONIA DA INTERFACE DE K7

Para testar a interface do K7 é necessário primeiro construir o circuito da figura 5.16. Use um freqüencímetro para saber se a entrada do CI 1 pino 5 é 4800 Hz.

Figura 5.17 Receptor para gravador de áudios de 300 bps.

Figura 5.18 Fluxograma de leitura do K7.

Figura 5.19 Fluxograma de escrita do K7.

Sem estar com o UART no lugar a freqüência no pino 1 do CI 2b deve ser de 2400 Hz. Aterrando o pino 1 do CI 2b, a saída deve mudar para 1200 Hz. Em ambos os casos, as tensões de 1V e 0,1 devem aparecer na entrada auxiliar e do microfone respectivamente.

O receptor usa estas freqüências geradas para que façamos seu ajuste. Com a saída em 2400 Hz faça uma ligação da saída da interface para a entrada do receptor (figura 5.17). Usando um osciloscópio verifique que a forma de onda no pino 6 do CI 1 é uma onda quadrada de 2400 Hz.

Agora com o osciloscópio no pino 6 do CI 2 ajuste R1 para que a tensão neste ponto seja máxima. Colocando a ponta do osciloscópio no pino 6 do CI 3 e mudando a freqüência para 1200 Hz repita o procedimento ajustando R2.

O potenciômetro R3 ajusta o ponto no qual o comparador chaveia entre os níveis lógicos quando da mudança de freqüência. A maneira certa de se ajustar é usando um gerador de função na entrada e colocando R3 para chavear exatamente em 1800 Hz. O resultado deve ser um nível lógico limpo quando da mudança das freqüências de entrada entre 1200 Hz e 2400 Hz.

O ajuste do comparador não é crítico.

CAPÍTULO 6

O SOFTWARE DO MONITOR

A função de um sistema operacional é prover o programador com um conjunto de ferramentas para ajudá-lo a desenvolver, depurar e executar um programa. Em geral o sistema operacional ajuda o programador a controlar os recursos do computador, e a eliminar seu envolvimento com as manipulações repetitivas dos códigos de máquina. Sistemas operacionais abrangem um largo espectro de complexidade. Sistemas pequenos, por exemplo, provêm do programador uma maneira rudimentar de escrever e ler dados da memória; em sistemas grandes, por outro lado, pode-se remanejar dinamicamente posições de memória e periféricos.

Sistemas grandes alocam recursos do computador para mais de um usuário em multiprogramação, multitarefa ou partição de tempo. Um sistema desta grandeza ultrapassa a capacidade do computador descrito neste livro. Sendo este o caso, qual seria o sistema operacional ideal para o PAZ? Como visto anteriormente, o objetivo de um sistema operacional é controlar os recursos do computador. O computador PAZ descrito nos capítulos anteriores contém os seguintes recursos:

- Microprocessador Z80
- 1204 bytes de memória EPROM
- 1024 bytes de memória programável (2048 opcional)
- Teclado de 19 teclas
- Mostrador de dados de dois caracteres
- Mostrador de endereços de quatro caracteres
- E/S serial (UART)

O sistema operacional deve prover acesso a esses recursos e dar ao usuário uma maneira de manejá-los durante a execução de programas. O sistema operacional projetado para o PAZ incluirá as seguintes facilidades e funções:

- Partida fria
- Partida quente
- Mostrar e trocar o conteúdo da memória
- Mostrar e trocar o conteúdo dos registradores
- Executar (começar a execução de um programa em um ponto determinado)
- Entrada e saída serial

Cada um será explicado em detalhes com relação às suas funções e implementação do programa.

1. Funções do Sistema Operacional

Operação de partida fria

O sistema operacional deve estar pronto imediatamente após o equipamento ser ligado. No passado, alguns sistemas proviam esta capacidade armazenando em ROM uma pequena rotina de carga (BOOTSTRAP). Esta rotina de carga era, então, usada para carregar o sistema operacional na memória, através de um outro periférico, como, por exemplo, de uma leitora de fita de papel ou de um K7. O avanço da tecnologia eliminou este passo. O sistema operacional do seu computador reside permanentemente em EPROM, e está pronto para ser executado assim que o equipamento é ligado ou o botão de RESET é apertado. Ao se apertar o botão de RESET coloca-se o PC (contador de programa) do Z80 com zero.

Com o próximo ciclo de máquina o processador começa a execução da instrução localizada em 00_{16} na memória. O sistema operacional do Z80 começa a execução das instruções. Esta série particular de instruções de programa constituem a "partida fria", que estabelece as condições iniciais para o sistema operacional.

O sistema operacional inicializa, então, o apontador da pilha (SP) para uma área na memória programável para manter as operações com a pilha. Esta pilha é necessária para a execução das instruções de RESTART e CALL. Se não inicializarmos a pilha antes da execução de uma instrução CALL ou RESTART, os efeitos da execução seriam imprevisíveis. Neste sistema operacional o apontador da pilha é colocado para o endereço $07C4_{16}$ da memória.

Operação de partida quente

Depois da inicialização do SP, o sistema operacional entra no módulo de reconhecimento de comando. Antes de discutirmos esta função do sistema operacional, algumas outras funções devem ser explicadas. O Z80 oferece ao usuário oito instruções de RESTART vetoradas (veja o capítulo 3 para a descrição das instruções). Por exemplo, a execução de um RST 08_{16} faz com que o PC seja armazenado na pilha e a execução do programa começará no endereço 08_{16}.

As instruções de RESTART a seguir são obtidas do sistema operacional.

RST 10_{16}
RST 18_{16}
RST 20_{16}
RST 28_{16}
RST 30_{16}
RST 38_{16}

A execução de qualquer uma dessas instruções faz com que o sistema operacional pule para uma posição da memória programável. Nesta posição o usuário executa uma instrução de pulo (JUMP) para vetorar o computador para uma nova posição.

Os RST 00_{16} e RST 08_{16} foram reservados pelo sistema operacional para funções especiais e não farão um JUMP para uma posição da memória programável. Estas duas instruções de RST podem ser usadas na depuração de programas. O RST 00_{16} fará a mesma função que o botão de RESET. A execução de um RST 00_{16} pelo Z80 resulta em uma "partida quente". Este módulo salva os dados existentes nos registradores na "área salva de registradores" localizada na memória programável (veja a lista do sistema operacional no apêndice D). O módulo irá tirar também da pilha o endereço de retorno do usuário e salvá-lo na área salva de registradores. O sistema operacional então entra no modo de reconhecimento de comando para esperar pelo próximo comando. O uso desta facilidade permite ao programador salvar os registros, o apontador, os flags e o contador de programa, antes de usar qualquer facilidade de depuração do sistema operacional. Uma descrição detalhada do módulo "partida quente" é dado na seção II.2 deste capítulo.

Desenvolvimento do programa e rotinas de depuração

As rotinas de partida fria e partida quente seguem a seqüência de comandos de entrada. Com estas rotinas de comando o programador é capaz de examinar e trocar dados na memória ou registradores, e de iniciar a execução em uma posição específica do programa na memória. Na entrada do módulo de comando de entrada, o sistema operacional mostra "FFFF" na seção de endereçamento, e "FF" na seção de dados do mostrador hexadecimal. O

usuário então implementa uma das três funções de comando apertando a tecla de SHIFT e as teclas "0", "1" ou "2". Um "SHIFT 0" (as teclas SHIFT e 0 apertados simultaneamente) diz ao sistema operacional para entrar no modo de mostrar e trocar os dados da memória, o "SHIFT 1" entra no modo de mostrar e trocar os dados dos registradores, e um "SHIFT 2" entra no modo de execução.

Amostragem e troca da memória

A função de mostrar e trocar os dados da memória permite ao usuário examinar o conteúdo tanto da EPROM como da RAM. Durante a operação o endereço e o conteúdo são mostrados respectivamente nos mostradores. Esta função é iniciada executando-se um "SHIFT 0" quando o sistema está no modo de reconhecimento de comando (mostrador de endereço = FFFF e mostrador de dados = FF). Por este tempo, o sistema operacional está esperando, pelo usuário, para entrar um endereço de um a quatro dígitos hexadecimal pelo teclado; estes dígitos entram seqüencialmente. Se entram mais de quatro dígitos somente os últimos quatro dígitos (mostrados no mostrador) serão usados como endereço. A entrada desse endereço é realizada ao se pressionar a tecla de "NEXT". Isto faz com que o conteúdo deste endereço apareça nos dígitos de dados. Se o usuário quiser mostrar o conteúdo dos endereços subseqüentes, ele precisará somente continuar a apertar a tecla de "NEXT". Se o usuário quiser trocar o conteúdo de uma posição de memória mostrada, ele poderá entrar um novo dado apertando um valor de duas posições antes de apertar a tecla de "NEXT". Este valor é, então, carregado na memória quando a tecla de "NEXT" for apertada. Apertando a tecla de "NEXT" continua a amostragem seqüencial dos endereços e dados.

O término desta função é conseguido apertando-se as teclas de "RESET" ou "EXEC". O controle então retorna à parte do sistema operacional de reconhecimento de comando.

Exemplo de amostragem da memória

Tecla	Endereço	Dado
	FFFF	FF
"SHIFT 0"	0000	FF
1	0001	FF
A	001A	FF
F	01AF	FF
"NEXT"	01AF	01
"NEXT"	01B0	1C
"RESET"	FFFF	FF

Exemplo de troca do conteúdo da memória

Tecla	Endereço	Dado
	FFFF	FF
"SHIFT 0"	0000	FF
4	0004	FF
0	0040	FF
0	0400	FF
"NEXT"	0400	01
2	0400	02
1	0400	21
"NEXT"	0401	05
6	0401	06
A	0401	6A
"EXEC"		

Os resultados serão:	Endereço	Dado
	0400	21
	0401	6A

Amostragem e troca do conteúdo dos registradores

A função de mostrar e trocar o conteúdo dos registradores permite ao usuário examinar e trocar os conteúdos dos registradores, salvos do Z80. Isto é conseguido executando-se um RST 1 (partida quente) durante a execução de um programa. Durante a execução dessa função, o conteúdo dos registradores são mostrados no mostrador de endereços. Os registradores de 8 bits nos dois dígitos mais baixos do mostrador de endereços e os dois dígitos mais altos serão preenchidos com zeros. Um código que indica qual o registro que está sendo mostrado aparece nos dígitos de dados. A tabela 6.1 descreve os códigos dados aos registradores, bem como as teclas que iniciam uma amostragem particular dos registros.

Código (Mostrado no mostrador de dados)	Registrador Z80 (Mostrado no mostrador de endereço)	Tecla
02	IX	2
03	IY	3
04	SP	4
05	PC	5
06	I	6
07	R	7
08	L	8
09	H	9
0A	A	A
0B	B	B
0C	C	C
0D	D	D
0E	E	E
0F	F	F
40	L'	"SHIFT 0"
41	H'	"SHIFT 1"
42	A'	"SHIFT 2"
43	B'	"SHIFT 3"
44	C'	"SHIFT 4"
45	D'	"SHIFT 5"
46	E'	"SHIFT 6"
47	F'	"SHIFT 7"

Tabela 6.1 Código/Registro/Seqüência.

A função de mostrar e trocar os conteúdos dos registradores é iniciada apertando-se "SHIFT 1" quando o sistema está no modo de reconhecimento de comando.

Nesta hora o sistema operacional está esperando que se entre com um dígito correspondente ao código do registrador. Se mais de um dígito for apertado, somente o último será considerado quando a tecla "NEXT" for apertada. Para a amostragem dos registros subseqüentes basta pressionar a tecla "NEXT".

Para os registradores de 16 bits os últimos 4 dígitos apertados é que serão aceitos, no caso dos registros de 8 bits os últimos dois. O usuário termina esta função apertando a tecla de "EXEC". O controle volta ao sistema operacional no modo de reconhecimento de comando.

Execução ("EXEC")

A função de execução (EXEC) permite ao usuário trocar o conteúdo do PC (contador do programa) para que o Z80 execute as instruções que estão a partir de um endereço selecionado pelo usuário. A função EXEC é iniciada apertando-se "SHIFT 2" quando o sistema está no modo de reconhecimento de comando. Agora o usuário deve entrar com um endereço de um a quatro dígitos. A execução começa quando a tecla de NEXT ou EXEC é apertada. Isto faz com que os registros do Z80 sejam guardados na área salva de registros (veja apêndice D) e a execução comece no endereço especificado pelo usuário.

Exemplo de amostragem do conteúdo de um registro

Tecla	Mostrador de dado (Registrador)	Mostrador de endereço (Conteúdo do registrador)
	FF	FFFF
"SHIFT 1"	00	FFFF
A	0A	FFFF
"NEXT"	0A	005C
"NEXT"	0B	0063
"RESET"	FF	FFFF

Exemplo de troca do conteúdo de um registro

Tecla	Mostrador de dado (Código do registrador)	Mostrador de endereço (Conteúdo do registrador)
	FF	FFFF
"SHIFT 1"	00	FFFF
5	05	FFFF
"NEXT"	05	043A
4	05	0004
2	05	0042
C	05	042C
"NEXT"	06	00FF
"NEXT"	07	0003
"EXEC"		

Exemplo de "EXEC"

Tecla	Mostrador de Endereço	Mostrador de Dado
	FFFF	FF
"SHIFT 2"	0000	FF
1	0001	FF
A	001A	FF
C	01AC	FF
F	1ACF	FF
"NEXT" ou "EXEC"		

Rotinas de e/s serial

O computador PAZ inclui uma facilidade de E/S serial implementada com um UART. Esta interface permite a comunicação serial entre o computador e periféricos, tais como impressoras ou terminais de vídeo. Para ajudar o usuário utilizar esta facilidade, o sistema operacional tem um módulo de diagnóstico para o UART, um módulo de entrada serial e um módulo de saída serial. Os módulos de entrada e saída são sub-rotinas que podem ser chamadas durante a execução de um programa.

Módulo de diagnóstico do UART

Este módulo provê um meio de se verificar a operação do UART. Para se usar esta facilidade o usuário deve primeiro ligar as linhas de saída e entrada serial juntas para que os dados enviados possam ser lidos pelo próprio UART. A sub-rotina de diagnóstico é iniciada usando-se a função EXEC. A execução começa em $032D_{16}$. Uma vez iniciado, o módulo de diagnóstico (UATST) envia dados para o UART e espera que o dado esteja à disposição. O *status* do UART é verificado para ver se não ocorreu alguma falha. Se ocorrer alguma falha o *status* do UART é então mostrado

nos dois dígitos mais baixos do mostrador de endereço (veja a tabela 6.2 para código de erros). Se não existirem erros, o dado é lido e mostrado nos dois dígitos do mostrador de dados. Uma comparação é feita entre o dado enviado e o recebido. Se os dois bytes forem iguais, o caracter enviado é incrementado e outro byte é enviado. O programa continua até que a tecla de RESET seja apertada ou até que um erro seja detectado. Se o dado enviado não for igual ao recebido, $0F_{16}$ será mostrado nos dois dígitos menos significativos do mostrador de endereços e o programa parará. A figura 6.1 mostra o fluxograma desta rotina.

Código mostrado	Erro
12_{16} ou 13_{16}	Erro de Paridade
$0A_{16}$ ou $0B_{16}$	Erro de Tamanho (Framing)
06_{16} ou 07_{16}	Erro de Sobreposição (Overrun)
00	Buffer de Transmissão não está vazio
$0F_{16}$	Caracter enviado ≠ Caracter recebido

Tabela 6.2 Códigos de erro do UART.

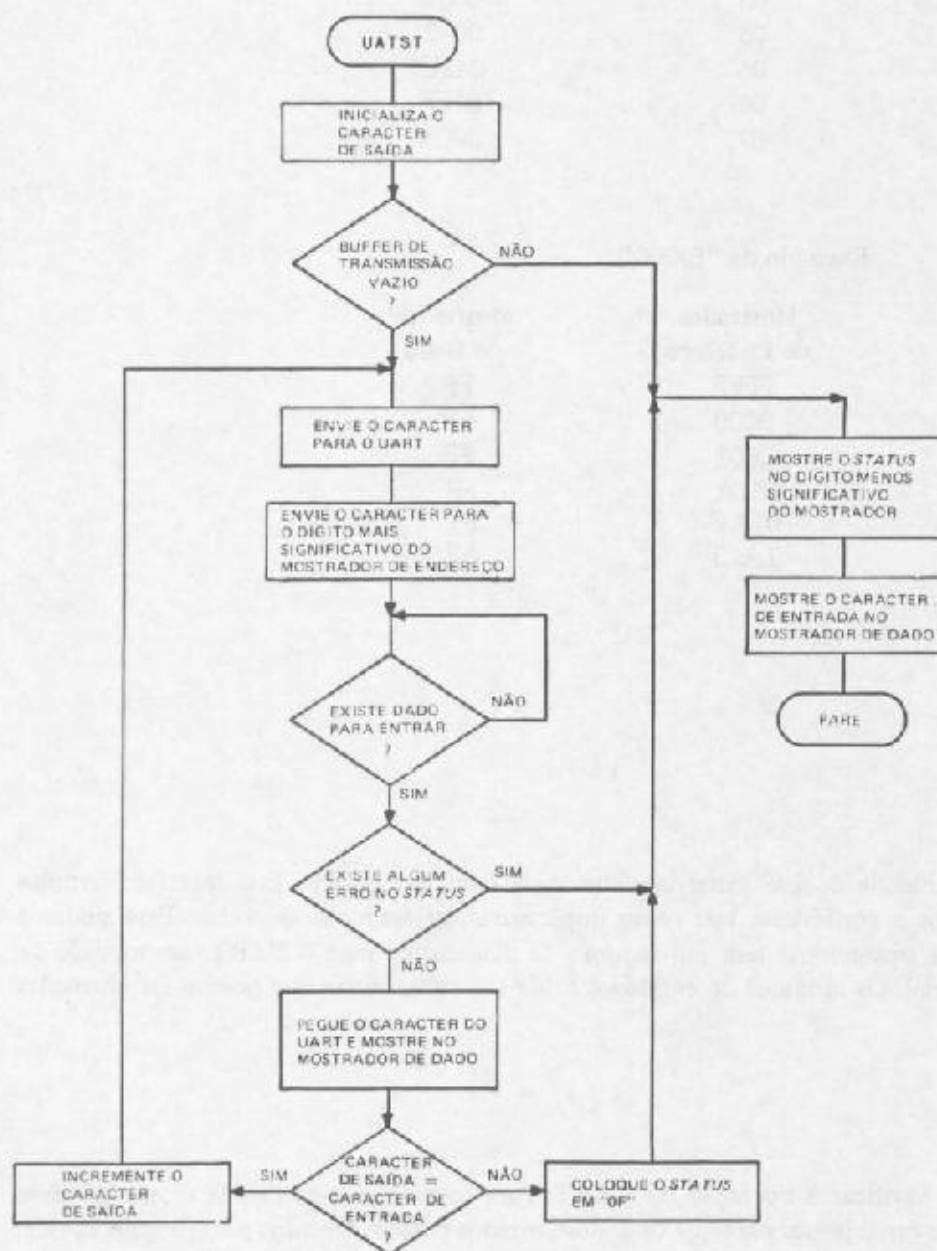



Figura 6.1 Fluxograma do módulo de diagnóstico.

Módulo de entrada serial

O módulo de entrada serial foi incluído para que o usuário pudesse ler dados serial de dispositivos externos. Para valer-se desta facilidade, o usuário deve alocar uma área de memória onde os dados recebidos serão guardados e designar o número de caracteres que serão recebidos. O endereço do buffer de entrada é armazenado no endereço $07F9_{16}$ na memória (veja apêndice D), e o número de caracteres é guardado no endereço $07FD_{16}$. A recepção da comunicação começa quando o módulo TTYINP for chamado.

Exemplo de iniciação da entrada serial

```
TTYINP  EQU  035F16        Endereço do Módulo de Entrada
BUFFER  EQU  07F916        Endereço do Buffer de Entrada
NCHAR   EQU  80            Número de Caracteres a ser Recebido
TTYIBU  EQU  07F916        Endereço do Sistema Operacional
TTYIC   EQU  07FD16        Endereço do Sistema Operacional
        LD  HL, BUFFER     Inicializa Buffer para o Sistema Operacional
        LD  (TTYIBU), HL
LD  A, NCHAR               Inicializa o Contador de Caracteres
LD  (TTYIC), A
CALL TTYINP                Chama  a Rotina de Entrada Serial do UART
```

O dado lido pelo módulo de entrada serial será guardado no buffer especificado pelo usuário até que a seqüência seja terminada. Quando isto ocorrer, o controle retorna ao programa do usuário na próxima instrução. O término do processo de entrada pode ser afetado pelas seguintes condições:

- Erro de *status*
- Número de caracteres lidos igual ao número de caracteres determinados
- Recebimento de um caracter de comando (VOLTA DO CARRO) carriage return (ASCII $0D_{16}$)

Se um erro de *status* for detectado, o registro A (acumulador) será igual a 80_{16}, quando o controle retornar ao usuário. Se o término for devido ao preenchimento do buffer corretamente o registro A conterá o valor de 00_{16}. Entretanto se o término for devido a um comando de retorno de carro, o registro A será igual ao número de caracteres que faltam para entrar. A figura 6.2 mostra o fluxo lógico do módulo TTYINP.

Módulo de saída serial

O módulo de saída serial é feito de maneira que o usuário seja facilitado no envio de dados para dispositivos externos. Para usar este módulo o operador designa um endereço para o buffer de saída de dados e o número de caracteres a serem transmitidos. O endereço do buffer de saída deve ser armazenado em $07FB_{16}$ na memória (veja o apêndice D) e o número de caracteres a serem enviados deve ser guardado no endereço $07FE_{16}$. A transmissão dos dados começa quando a rotina TTYOUT for chamada.

Exemplo de rotina de saída de dados

```
TTYOUT  EQU  039E16        Endereço do módulo de saída
BUFFER  EQU  07FB16        Endereço do buffer de saída
NCHAR   EQU  35            Número de caracteres a ser transmitido
TTYOBF  EQU  07FB16        Endereço do sistema operacional
TTYOC   EQU  07FE16
        LD  HL, BUFFER     Endereço do buffer para o sistema operacional
        LD  (TTYOBF), HL
        LD  A, NCHAR       Contador de caracter para o sistema operacional
        CALL TTYOUT        Chama a rotina de saída serial
```

O controle retornará ao usuário quando:

- O buffer de saída estiver vazio
- O buffer de transmissão não estiver à disposição, indicando um erro.

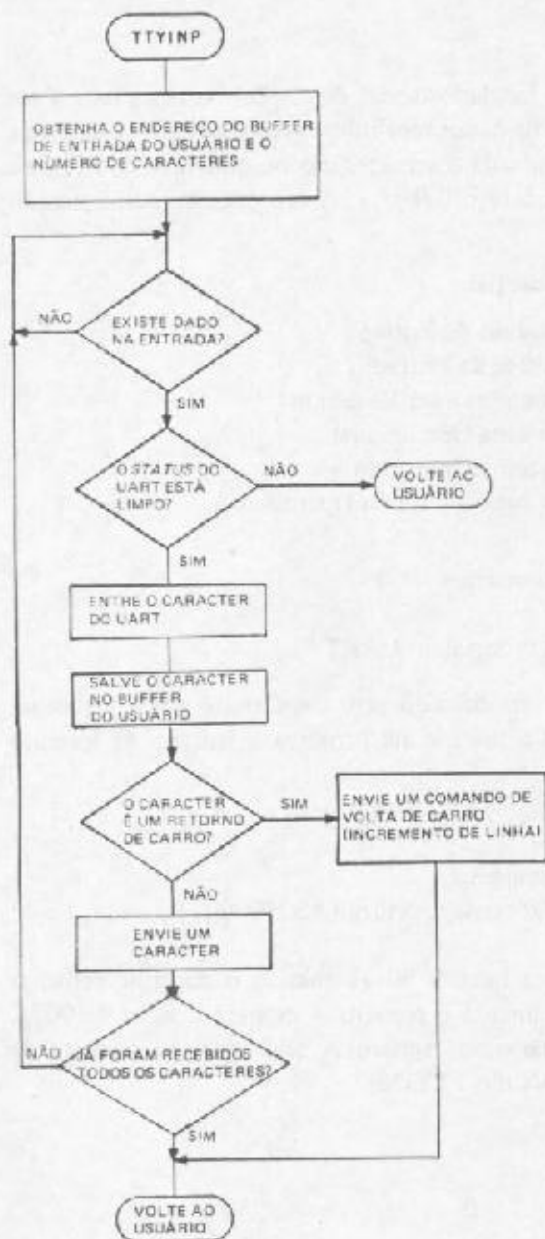



Figura 6.2 Fluxograma do módulo de entrada serial.

Se ocorrer um término normal, o registrador A conterá 00_{16} quando o controle retornar ao usuário. Entretanto, se ocorrer um término prematuro, o registro A conterá 01_{16}. A figura 6.3 mostra o fluxo lógico do módulo de saída.

II. Descrição do módulo do sistema operacional

II.1. Módulo de partida quente

O módulo de partida quente (WARM1) é responsável por salvar todos os registradores do Z80, na área de salva de registros localizada na porção reservada da memória programável. Ao entrar no módulo, os registros A, H e L serão salvos para que o módulo possa usar esses registros quando da sua execução. O próximo passo é salvar o PC do usuário que está na pilha e colocá-lo em uma posição de memória.

O par de registradores AF é colocado na pilha e depois removido para o par de registros HL. Isto faz com que o registro de FLAG possa ser colocado na área de salva de registros. O resto dos registradores são salvos na área de salva de registros. Ao completar esta tarefa o módulo retorna o comando ao módulo de reconhecimento de comando (veja o apêndice D para maiores detalhes). A figura 6.4 mostra o fluxograma do módulo de partida quente.

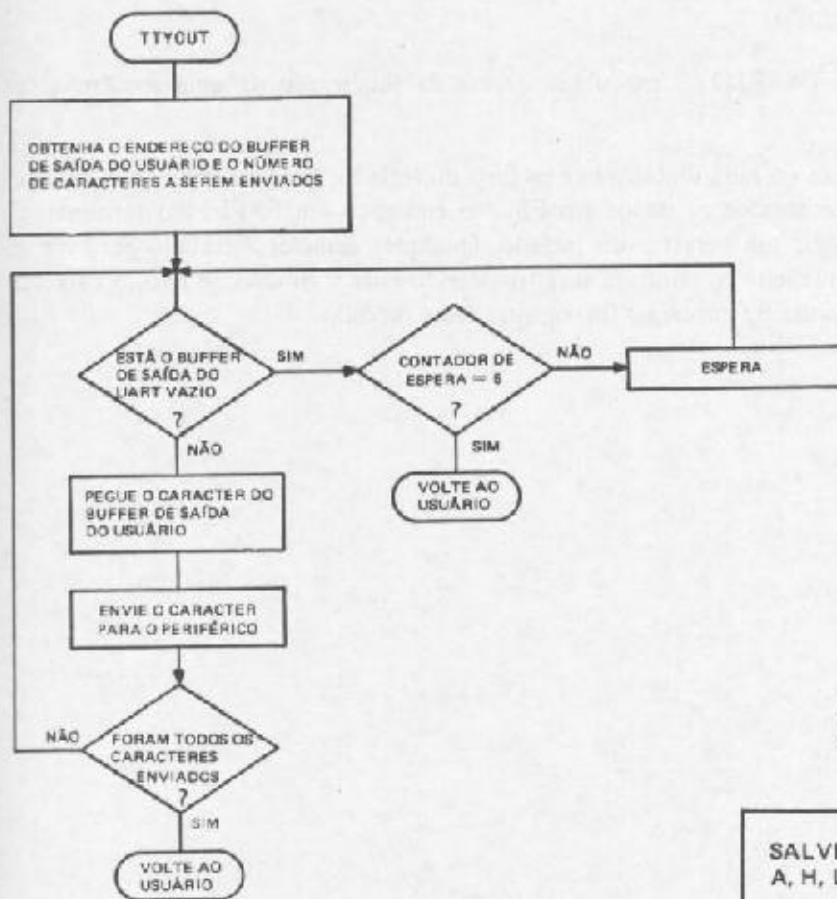



Figura 6.3 Fluxograma do módulo de saída.

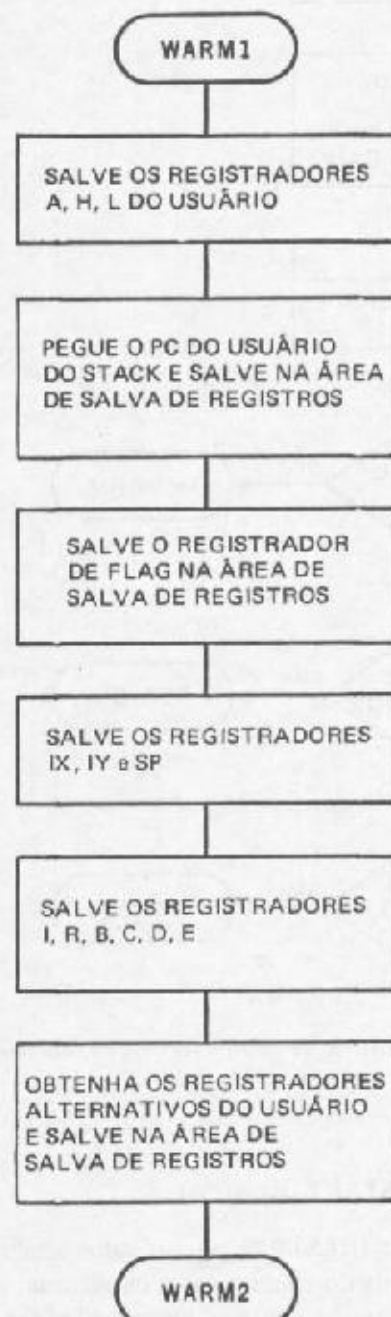



Figura 6.4 Fluxograma do módulo de partida quente (WARM1).

II.2. **Módulo de reconhecimento de comando**

O módulo de reconhecimento de comando (WARM2) é executado depois da finalização de uma seqüência de partida fria.

Quando iniciado, o módulo limpa o buffer de entrada do teclado e os flags do teclado. Isto remove a ambiguidade para operações futuras. O módulo coloca no mostrador os dados em FF e o endereço em FFFF. Ao terminar, o módulo entra na sub-rotina de KEYIN para pegar um caracter do teclado. Qualquer caracter é testado para ver se corresponde a uma das três funções. Se for verdadeiro, o controle será transferido para a função, se não, o caracter será ignorado e o módulo esperará por outro. A figura 6.5 mostra o fluxograma deste módulo.

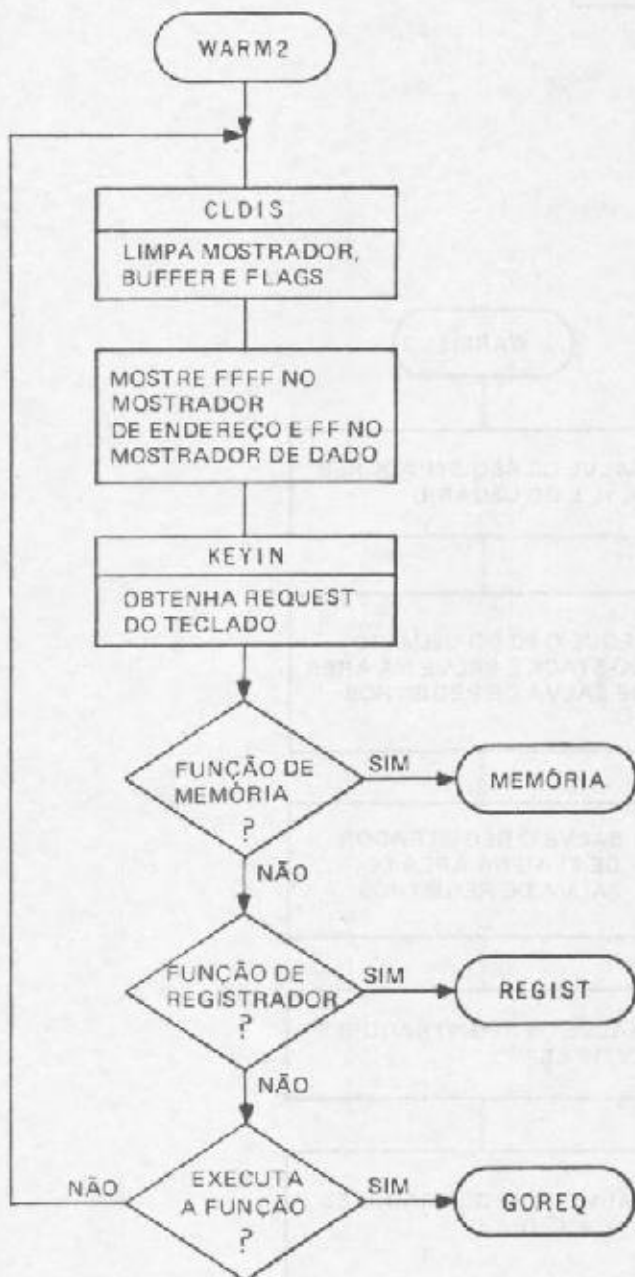



Figura 6.5 Fluxograma do módulo de reconhecimento de comando.

II.3. **Módulo de RESTART (Reinício)**

O módulo de restart (RESTRT) pega o valor armazenado na área salva da memória programável; logo restaura os registros de 8 e 16 bits do usuário antes de retornar o controle para a posição especificada pela área salva do PC. Este procedimento restaura os registros alterados, e depois os registros de trabalho. Em qualquer caso, os registros

de flag são restaurados pela retirada do dado da pilha e colocado, então, no registro F. A fim de sair para o endereço de restart do usuário, o PC salvo é colocado na pilha e um "RET" (instrução de retorno) é executado (veja apêndice D para detalhes adicionais). A figura 6.6 mostra o fluxo lógico do módulo de RESTART.

II.4. Módulo de entrada de teclado

O módulo de entrada de teclado (KEYIN) fornece a interface primária entre o computador e o usuário. Inicialmente, este começa a ler o dado a partir da porta de entrada do teclado; ficando em um loop, verificando o MSB (bit mais significativo) do dado. O MSB é o strobe de tecla apertada. Quando este vai para um nível lógico um, os sete LSBs (bits menos significativos) da porta de entrada do teclado são guardados como caracter de entrada desejado. O módulo, então, retorna ao programa do usuário com o caracter do teclado no acumulador (veja apêndice D para detalhes adicionais). A figura 6.7 mostra o fluxo lógico do módulo de entrada do teclado.

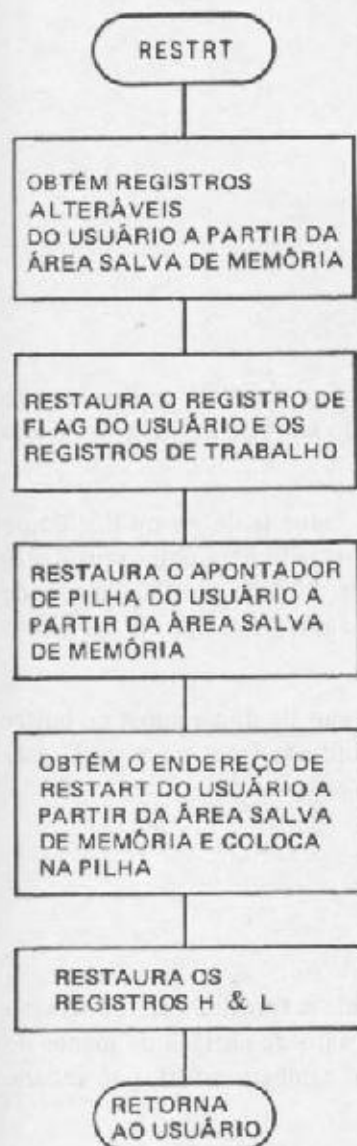



Figura 6.6 Fluxograma do módulo de restart (RESTRT).

Figura 6.7 Fluxograma do módulo de entrada do teclado.

II.5. Módulo de entrada de um caracter

A função deste módulo (ONECAR) é dar entrada a um ou mais caracteres do teclado. Este módulo também indica o último caracter e se este foi acompanhado por uma tecla "NEXT" ou "EXEC".

No início, o buffer de entrada e os flags do teclado são limpos. (O mostrador de dados pode ou não ser limpo dependendo das necessidades do módulo chamado.) Um módulo espera por um caracter de entrada a ser passado. Quando este recebe um caracter, verifica se este é um "NEXT", "EXEC", ou um dado válido. No evento de entrada ser um "NEXT" ou "EXEC", o flag de teclado apropriado é ligado de acordo com o flag de não dados e o controle retorna para o usuário (veja figura 6.8).

Se um caracter de dado não válido é recebido, um módulo é reiniciado. Na recepção de um dado válido, o dado é armazenado em um buffer de entrada de 1 byte, e o módulo espera pelo próximo caracter de entrada. Este caracter é processado de maneira similar ao já descrito com a seguinte exceção: no evento do caracter de entrada ser um "NEXT" ou "EXEC", somente o flag apropriado é ligado antes de retornar o controle para o usuário (veja apêndice D para detalhes adicionais). A figura 6.9 mostra o fluxo lógico do módulo de entrada de um caracter.

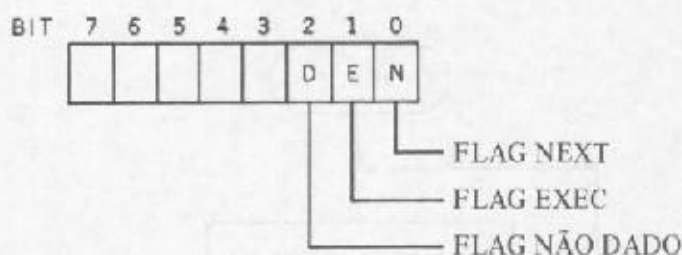



Figura 6.8 Configuração dos flags do teclado.

II.6. Módulo de entrada de dois caracteres

A função deste módulo (TWOCAR) é dar entrada a um ou mais caracteres do teclado e transferir para o usuário os últimos dois caracteres quando uma tecla "NEXT" ou "EXEC" for apertada. O módulo também notifica ao usuário o tipo da terminação que tomou posição.

Na entrada, o buffer de entrada e os flags de entrada são limpos. (O display de dados pode ser ou não limpo dependendo da necessidade do módulo chamado.) Este módulo chama o módulo de entrada para obter seus dados de entrada. O primeiro caracter é verificado para determinar se este é um "NEXT" ou "EXEC"; o flag de teclado apropriado é ligado de acordo com o flag de não dados; e o controle retorna para o usuário (veja figura 6.8). Se um dado não válido é recebido, o módulo é reiniciado.

A recepção de um dado válido fará com que o módulo formate o dado como um valor de dois dígitos no buffer de entrada do teclado. Então o controle retorna para o usuário com o flag ligado apropriado (veja apêndice D para detalhes adicionais). A figura 6.10 mostra o fluxo lógico do módulo de entrada de dois caracteres.

II.7. Módulo de entrada em quatro caracteres

A função deste módulo (FORCAR) é dar entrada a um ou mais caracteres do teclado e transferir para o usuário os últimos quatro caracteres quando uma tecla "NEXT" ou "EXEC" é apertada. No evento de entrada de menos do que quatro caracteres, os dígitos de maior ordem serão colocados em zero. O módulo também notifica ao usuário através dos flags de teclado (veja figura 6.8).

A operação deste módulo é muito similar ao do módulo de entrada de dois caracteres. A principal diferença recai na forma na qual o novo dado (entrada do teclado) é unido com o dado de entrada anterior do teclado (veja apêndice D para detalhes adicionais). A figura 6.11 mostra o fluxo lógico do módulo de entrada de quatro caracteres.

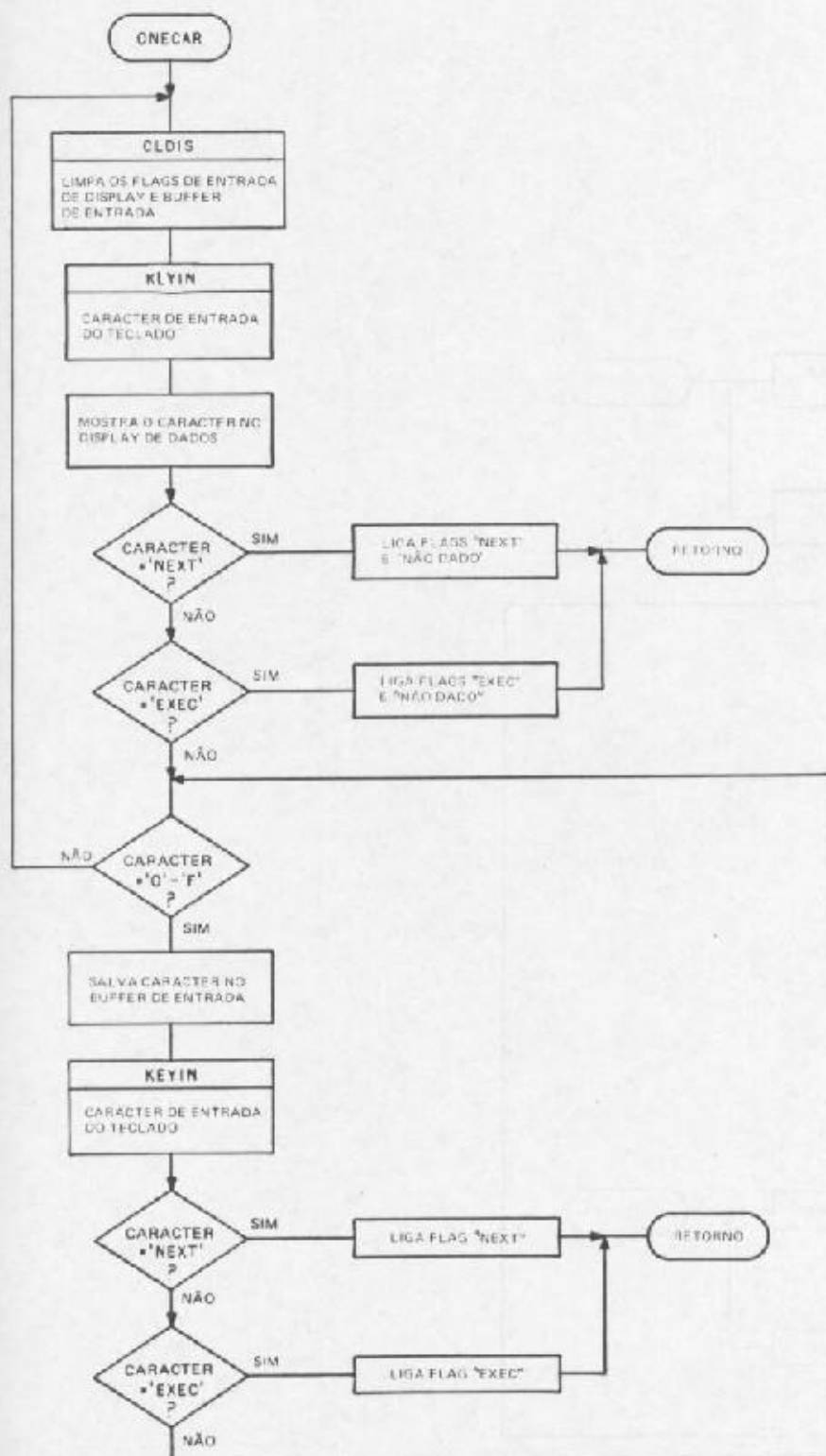



Figura 6.9 Fluxograma do módulo de entrada de um caracter (ONECAR).

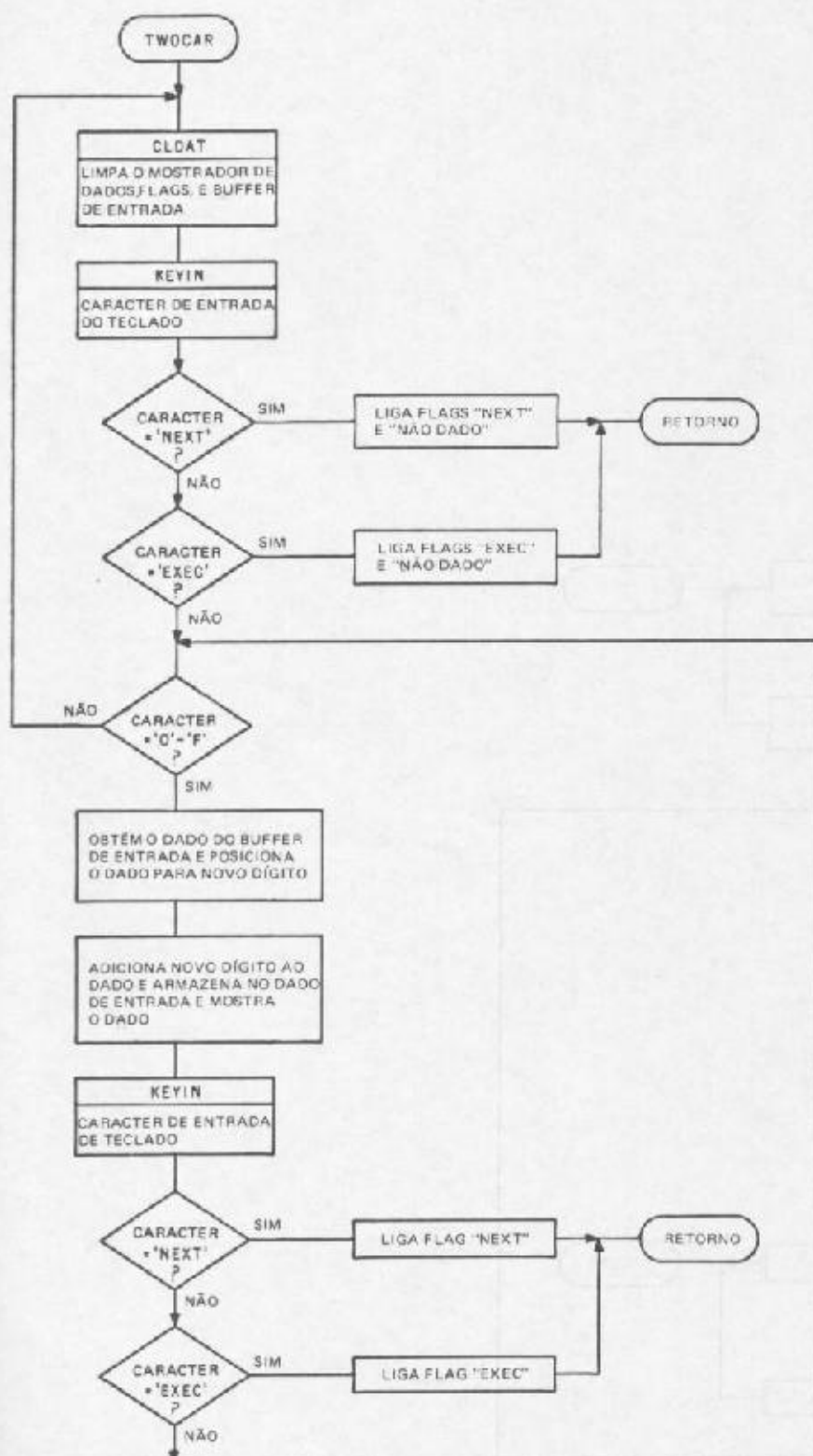



Figura 6.10 Fluxograma do módulo de entrada de dois caracteres (TWOCAR).

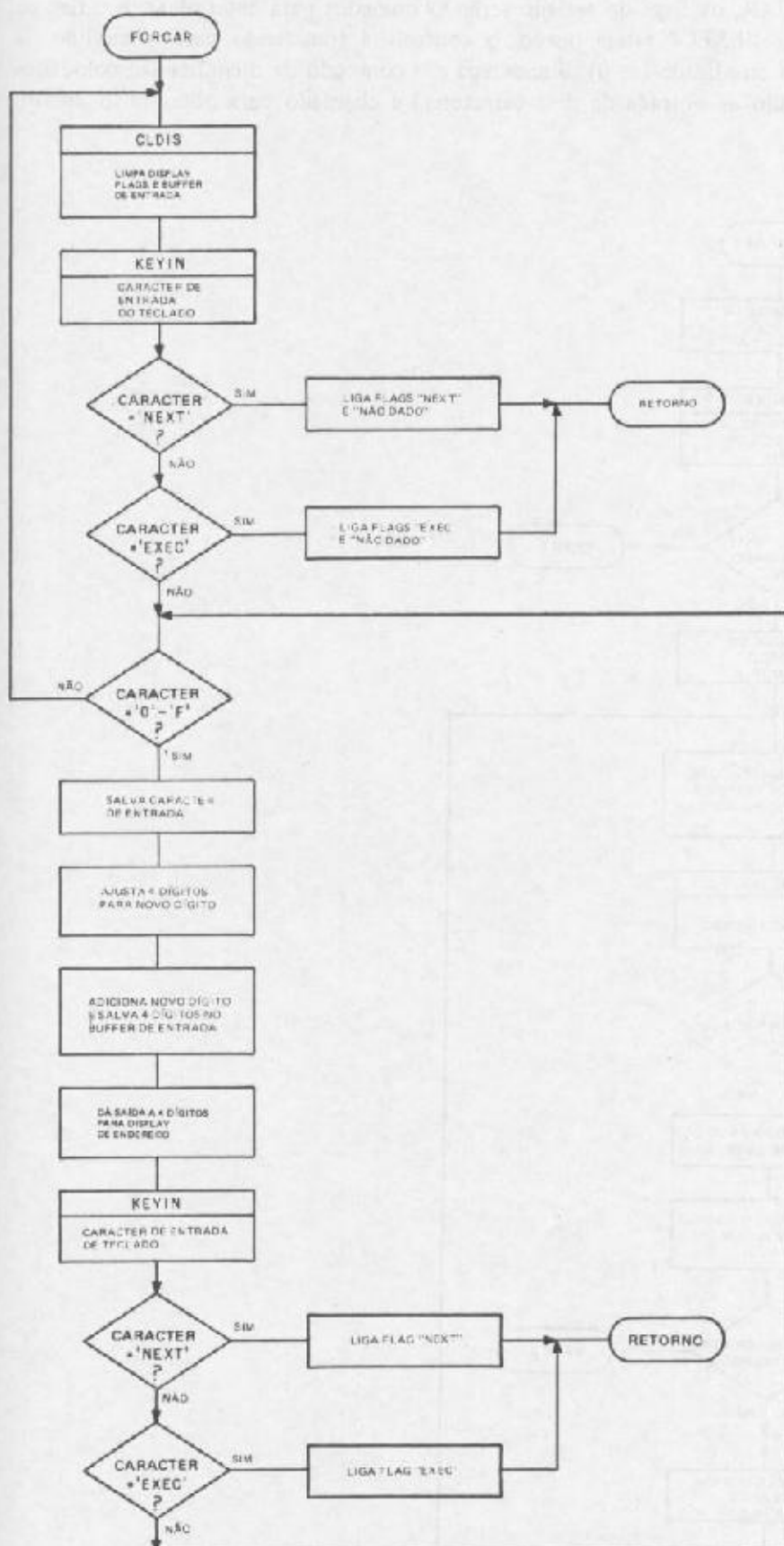



Figura 6.11 Fluxograma do módulo de entrada de quatro caracteres (FORCAR).

II.8. Módulo de display de memória e reposição

A função de display de memória e reposição é um dos três maiores módulos do sistema operacional. Na entrada (veja módulo de reconhecimento de comando), este módulo (MEMORY) faz uma chamada ao FORCAR (módulo de entrada de quatro caracteres) para obter a base de endereço de memória, na qual começará a mostrar os conteúdos da memória. Quando este retornar do FORCAR, os flags de teclado serão examinados para determinar se o flag de "EXEC" está ligado (= 1). Caso o flag de "EXEC" esteja ligado, o controle é transferido para o módulo de restart (RESTRT). Se o flag de "EXEC" não está ligado (= 0), o endereço e o conteúdo da memória são colodados nos displays apropriados. O TWOCAR (módulo de entrada de dois caracteres) é chamado para obter novo dado a partir da posição de memória mostrada.

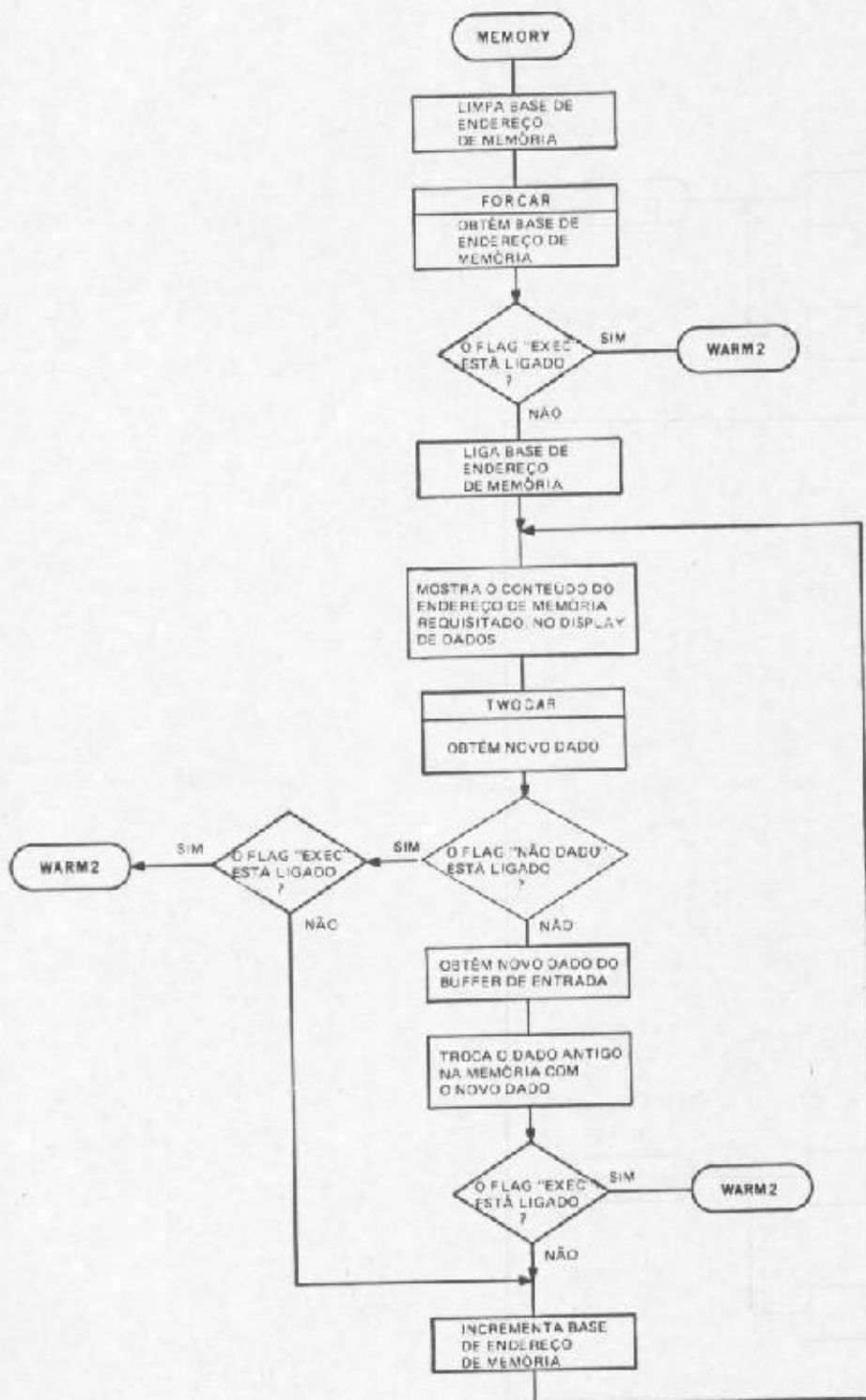



Figura 6.12 Fluxograma do módulo de display de memória e reposição (MEMORY).

Quando o controle retorna do TWOCAR, o módulo verifica o flag de "não dados". Se este flag está ligado (= 1), o flag de "EXEC" é examinado, e se estiver ligado, o controle é transferido para o módulo de reconhecimento de comando (WARM2). Se, por outro lado, o flag de "EXEC" estiver desligado (= 0), o endereço de memória do usuário será incrementado, mostrado no display de endereço, e seu conteúdo será mostrado no display de dados.

Se, no retorno do TWOCAR, o flag de "não dados" tiver desligado (= 0), o novo dado será extraído do buffer de entrada do teclado e armazenado na posição de memória mostrada. Nesta hora, o módulo determina se a saída do TWOCAR foi através de uma diretiva "EXEC" ou "NEXT". No caso do flag de "EXEC" estar ligado (= 1), o controle é transferido para o módulo de reconhecimento de comando (WARM2). Se, entretanto, o flag estiver desligado (= 0), o endereço de memória do usuário será incrementado, mostrado no display de endereços, e seu conteúdo será mostrado no display de dados. Então o módulo de entrada de dois caracteres é chamado para obter a próxima diretiva para o módulo de display de memória e reposição (veja apêndice D para detalhes adicionais). A figura 6.12 mostra o fluxo lógico do módulo de display de memória e reposição.

II.9. Módulo de display de registro e reposição

O módulo de display de registro e reposição (REGIST) é um dos três maiores módulos do sistema operacional. Este módulo chama o ONECAR (módulo de entrada de um caracter) para obter o código do registro de display inicial do usuário (veja tabela 6.1). No retorno do ONECAR, o flag de "EXEC" é verificado. Se este flag está ligado (= 1), o controle é transferido para o módulo de reconhecimento de comando (WARM2). Se o flag de "EXEC" está desligado (= 0), o índice do display do registro de base é calculado a partir do código do registro de display do usuário.

Nesta hora, o registro de index é verificado para ver se o registro requisitado é um registro de 8 ou 16 bits. Se o usuário requisita um registro de 16 bits, o código do registro é mostrado no display de dados, e o dado do registro requisitado é obtido da área de salva de registro e mostrado no display de endereços. O módulo então faz uma chamada ao FORCAR (módulo de entrada de quatro caracteres) para obter novo dado para o registro. No retorno, o flag de "não dados" é verificado. Se este flag está ligado e o flag de "EXEC" também está ligado, o controle é transferido para o RESTRT (módulo de reinício). Se os flags de "não dados" e "NEXT" estão ligados, o índice de registro do display é incrementado e mostrado no display de dados. O novo dado do registro é obtido da área de salva de registro e mostrado no display de endereços.

Se um registro de 8 bits foi solicitado, o código do registro (veja tabela 6.1) é mostrado no display de dados, e o dado é obtido da área de salva de registro e mostrado no display de endereços. Nesta hora, o módulo chama o TWOCAR para obter novo dado do registro mostrado. Quando o controle retorna do módulo de entrada de dois caracteres, o módulo determina o modo de execução através do exame dos flags do teclado. Se os flags de "não dados" e "EXEC" estão ligados, o controle é transferido para o módulo de reconhecimento de comando (WARM2). Se os flags de "não dados" e "NEXT" estão ligados, o registro de index é incrementado e o conteúdo do registro enviado para o display apropriado.

Se o flag de "não dados" está desligado, o novo dado do registro é obtido do buffer de entrada do teclado e armazenado na posição própria da área de salva de registro. Nesta hora, o flag de "EXEC" é verificado e, se ligado, o controle é transferido para o módulo de reconhecimento de comando (WARM2). Se o flag de "EXEC" está desligado, o dado do registro é mostrado e a diretiva do usuário é processada (veja apêndice D para detalhes adicionais). A figura 6.13 mostra o fluxo lógico do módulo de display de registro e reposição.

II.10 Módulo go execute

O módulo go execute (GOREQ) é o último das três maiores funções do sistema operacional. Na entrada (veja módulo de reconhecimento de comando), este módulo chama o FORCAR para obter o endereço de início de execução. No retorno do FORCAR, o flag de "não dados" é examinado para determinar o modo de execução. Se este flag está ligado (= 1), o controle é imediatamente transferido para o RESTRT. Este restaura os registros do Z80 recuperando a execução no endereço atual contido no PC a partir do buffer de entrada do teclado, e armazena na posição reservada para o PC dentro da área salva de registro. O controle é, então, transferido para o módulo de reconhecimento de comando (WARM2) o qual irá restaurar os registros com os dados salvados, e inicia a execução do programa do usuário no endereço especificado (veja apêndice D para detalhes adicionais). A figura 6.14 mostra o fluxo lógico do módulo go execute.

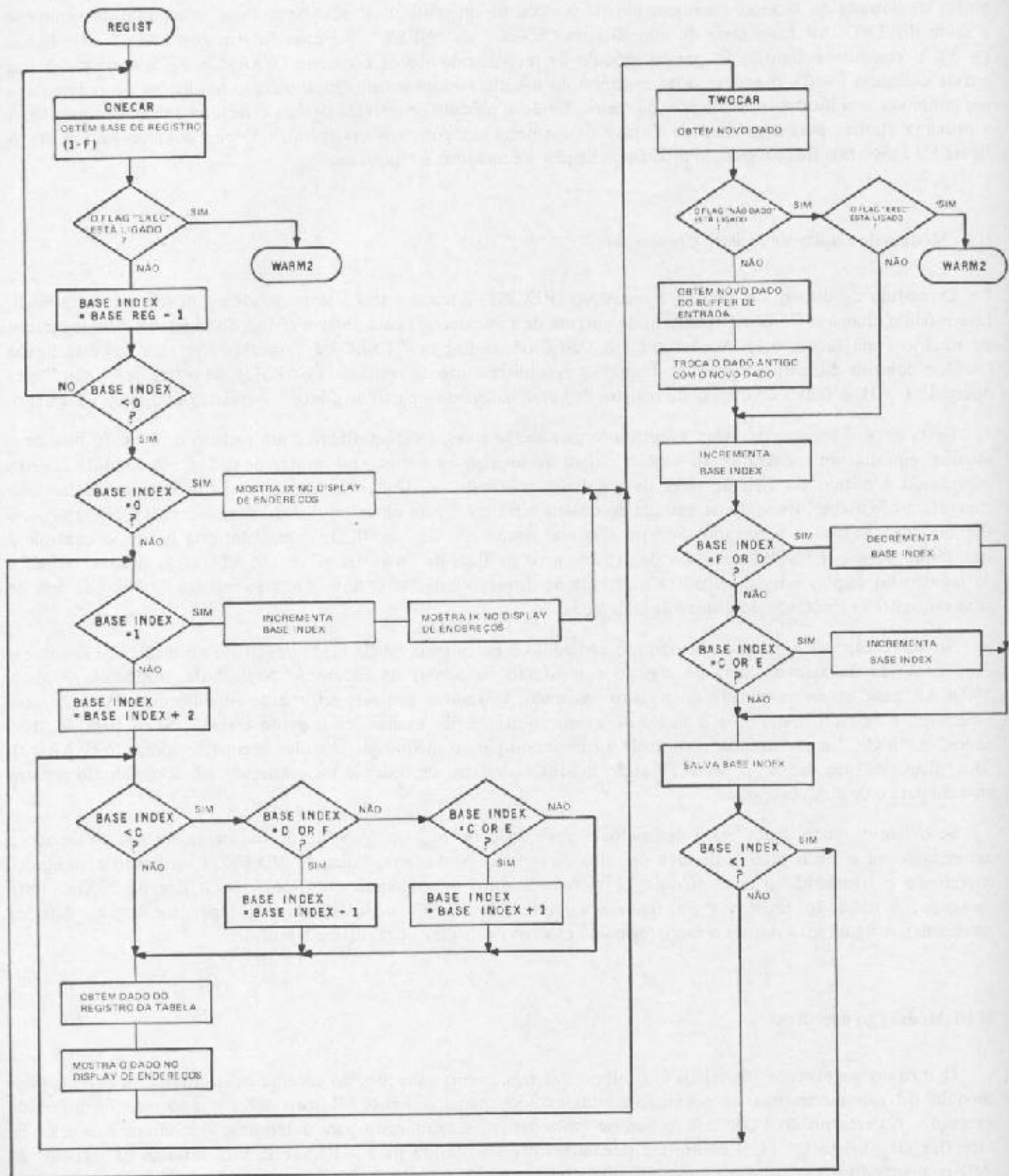



Figura 6.13 Fluxograma do módulo de display de registro e reposição (REGIST).

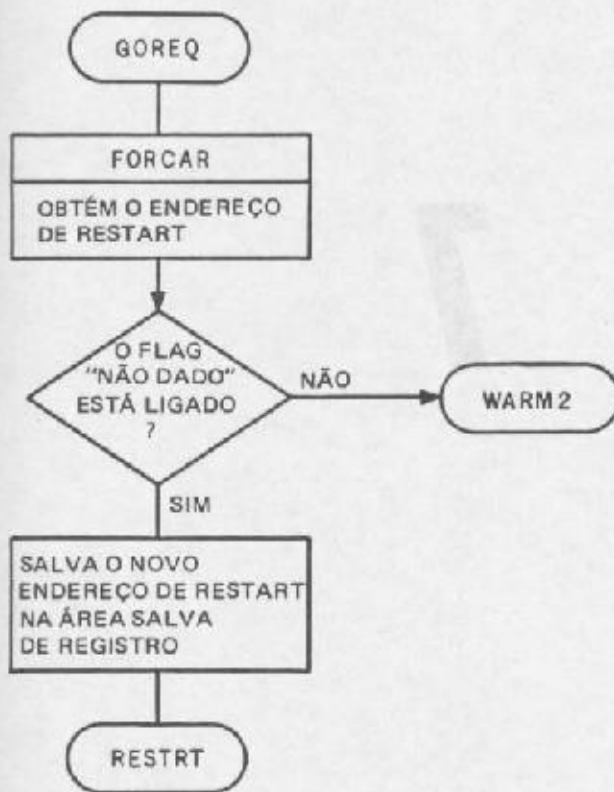



Figura 6.14 Fluxograma do módulo go execute (GOREQ).

CAPÍTULO 7

PROGRAMANDO UMA EPROM

O computador PAZ está sendo projetado de forma a não ser dispendioso, a ser realizável e fácil de construir. Para manter mínimo os custos e a complexidade, algumas características do computador que poderiam ajudar ao iniciante têm sido eliminadas. As mais visíveis dessas características são um painel frontal e um mostrador. Apesar de não influir de forma alguma na operação do computador, sua inclusão tornará mais fácil o teste final e o desenvolvimento de programas.

Para o teste próprio do PAZ, um programa deve estar na memória. Este programa não tem que ser muito longo — apenas umas poucas instruções são necessárias para determinar se o computador está funcionando. O problema aparece quando o usuário deseja rodar um programa de 50 ou 100 bytes de comprimento. Para efetivamente entrarmos com um código de máquina na memória programável do PAZ, um programa para coordenar esta atividade deve estar gravado em EPROM. Tal programa é chamado monitor e está descrito no capítulo 6.

Esta seção inclui informações de programação de EPROMs. Para resolver a situação de início descrevi um projeto para um par de programadores manuais de EPROMs. Carregar programas em um programador manual é tedioso. Eles são feitos para rotinas muito pequenas, tais como verificação de operação do sistema básico. Entretanto, uma unidade manual pode ser modificada para carregar o software monitor de 1K completo. Quando o PAZ estiver completamente operacional, você poderá usá-lo em conjunto com um programador automático, isto irá ajudar na escrita de EPROMs. No caso de você não desejar escrever sua própria EPROM, consulte o apêndice A para EPROMs programadas.

Uma rápida revisão de EPROMs

É sempre desejável ter a não volatibilidade de ROMs, como também a capacidade de ler/escrever das memórias programáveis. A EPROM é uma memória utilizada para leitura, é usada como uma ROM por extensos períodos de tempo, ocasionalmente é apagada e reprogramada quando necessário. O apagamento é permitido pela exposição do substrato do chip, coberto por uma janela transparente de quartzo, à luz ultravioleta. Falaremos sobre o apagamento no final deste capítulo.

O elemento de memória EPROM usado pela Intel e muitos outros fabricantes é um tipo de carga armazenada chamada transistor FAMOS (Floating-gate Alavanche injection Metal Oxide Semi-conductor). Aplicando-se seletivamente uma tensão de carga de 25V na célula endereçada, um determinado bit que constitui o programa

pode ser escrito na EPROM. Esta carga, devido ao material isolante, pode permanecer por anos. A exposição à luz ultravioleta intensa drena a carga e o resultado é o apagamento de toda informação programada.

Existem várias EPROMs no mercado –2708s, 2716s e 2732s são as principais. Para a maior parte, os montadores de computadores têm se afastado da dificuldade de programação das 1702s e têm optado pela maior facilidade de programação das 2708s e 2716s. Um benefício adicional é sua maior densidade de armazenamento. As EPROMs mais novas no mercado são consideravelmente mais caras do que a 2708. Considerando-se tudo isto, a 2708 passa a ser a melhor compra. Por estas razões, o programador de EPROM descrito neste capítulo é o 2708.

A figura 7.1 é o circuito para um programador manual 2708. O CI 5 e duas seções do CI 3 fornecem o pulso de +25V para a EPROM. O CI 5 está armado para uma duração de 1 ms e é gatilhado pela transição de 0 para 1 na sua entrada. A EPROM tanto fornece quanto requer corrente através da programação do pino 18. No modo escrita, quando CS/WE, pino 20, está em +12V e entre pulsos de programação, o pino 18 tem de ser puxado para baixo por um componente ativo porque este fornece uma corrente pequena. O pulso de programação é de cerca de 30 mA e não pode ser facilmente gerado sem o seguidor de emisssor Q1. Este pulso, no pino 18, deve estar entre 25 e 27V. Três baterias de 9V serão suficientes. (Uma alternativa é usar uma fonte de alimentação comercial de 24V, 50 mA. A fonte pode ser ajustada por resistor para produzir de 25 a 27V.)

Para escrever um byte na EPROM, um endereço de 10 bits, designando qual dos 1024 bytes receberá o dado, estará presente nas chaves de SW1 a SW10. Para iniciar na posição 0, todas as chaves estarão na posição fechada. Em seguida, os 8 bits que serão armazenados são colocados nas chaves de SW12 a SW19. Este byte de dados poderá ser mostrado no display de saída do LED 1 a LED 8. Finalmente, para o programador estar no modo de escrita, a chave SW 11 deve estar aberta. A inserção real do dado ocorre quando o botão de pulso de escrita PB1 é pressionado. Este fornece um pulso de 1 ms e 25V para o pino de programação da 2708. De acordo com as especificações do fabricante, nenhum pulso de programação deve ser maior que 1 ms. Para a máxima retenção de dados, 100 destes pulsos de programação são recomendados (totalizando 100 ms por byte).

Infelizmente, os 100 ms não podem ser aplicados de uma só vez. Os fabricantes especificam que isto deve ser feito seqüencialmente num total de 100 aplicações de 1 ms. Em resumo, isto significa que para um programa de 25 bytes, cada endereço deve ser escrito com um pulso e então repetir a operação até 100 vezes. Obviamente para uma retenção completa cada endereço deve ser reescrito em um programador automático.

A leitura do conteúdo armazenado na 2708 é facilmente executada no mesmo programador manual. Primeiro, todas as chaves de entrada de dado de SW12 a SW19 são abertas e então a chave "lê/escreve" SW11 é fechada (modo de leitura). Nenhum outro pulso é necessário. O display de saída mostrará o conteúdo do byte apontado pelas chaves de entrada de endereço de SW1 a SW10. Este permanecerá constante até que seja colocado um outro endereço. A leitura dos conteúdos pode ser feita pelo incremento destes 10 bits de endereço através da gama de endereços do programa.

Um programador um pouco mais complexo é demonstrado na figura 7.2. Três contadores programáveis são inseridos entre as chaves de entrada de endereço e a EPROM. Ao invés de mudar-se as posições das chaves para cada endereço, elas agora são usadas somente para armar os contadores em um endereço inicial. Se nós quisermos programar uma EPROM começando no endereço hexadecimal 3AA, as chaves deverão ser ligadas para este endereço e a chave de reset de endereço deverá ser pressionada. Os 10 LEDs, LEDA0 a LEDA9, lerão 3AA como o endereço. O dado a ser programado é colocado nas chaves de SW12 a SW19. Apertando-se o botão de escrita de dado PB1, o dado das chaves será armazenado. As posições de memória subseqüentes são programadas utilizando-se apenas as chaves de SW12 a SW19 e apertando-se PB1. Para se zerar o contador basta apertar o botão de clear (limpar).

É fácil ver como este programador manual facilita a leitura da memória. Coloque todas as chaves de entrada de dados no nível lógico 1 e a interface no modo leitura, selecione e carregue o endereço inicial. A leitura dos outros endereços é simplesmente uma operação repetida do botão de incremento de endereço.

Um programador automático

Você necessitará de um computador PAZ operacional para construir um programador automático. A complexidade do projeto pode ser reduzida consideravelmente utilizando-se as vantagens de decodificação dos strobes de E/S existentes no PAZ básico. O circuito mostrado na figura 7.3 utiliza menos 3 chips do que o programador manual da figura 7.2. Suas operações são similares, porém um pouco diferentes em detalhes.

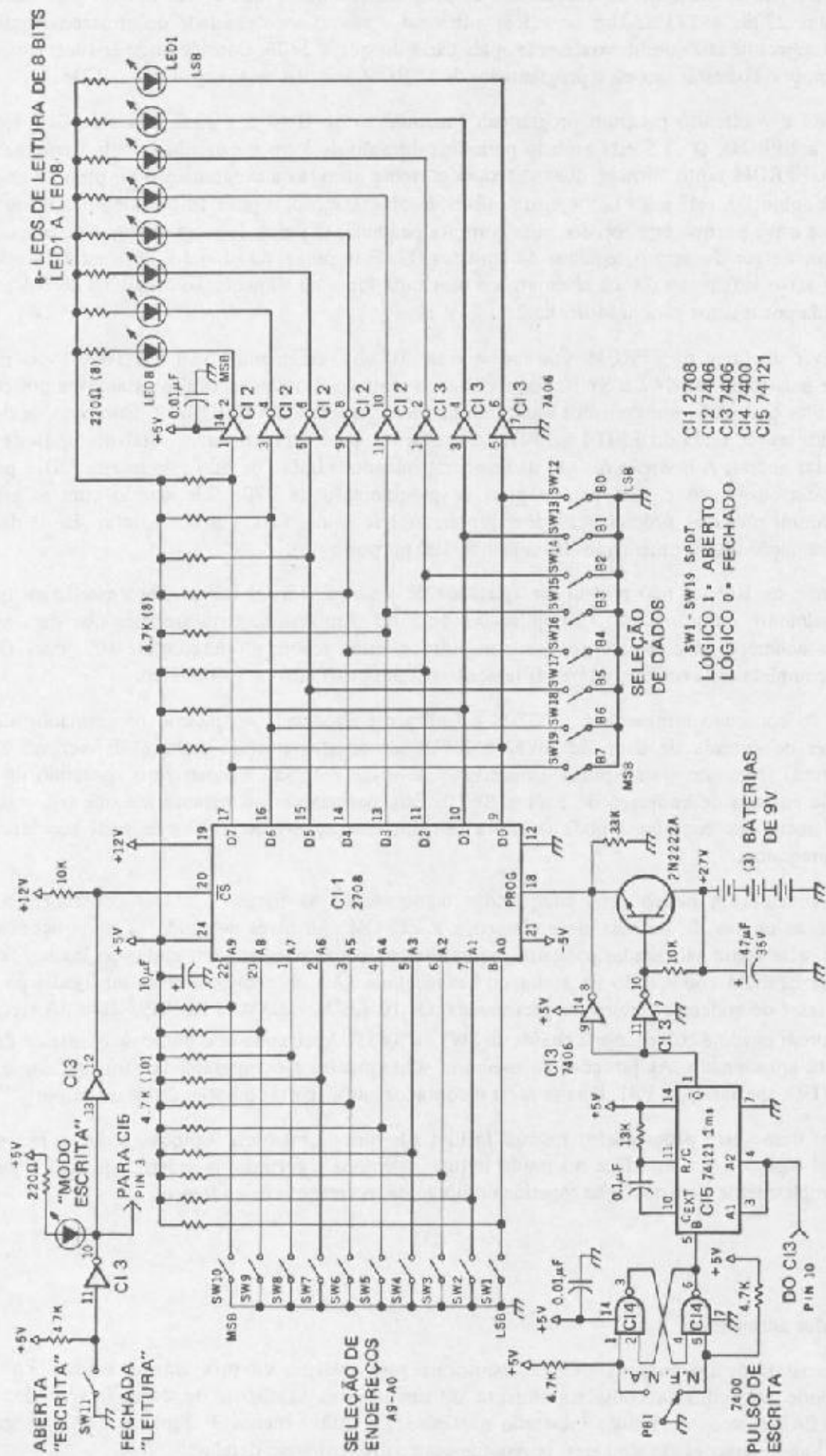



Figura 7.1 Diagrama esquemático de um programador manual para a 2708.

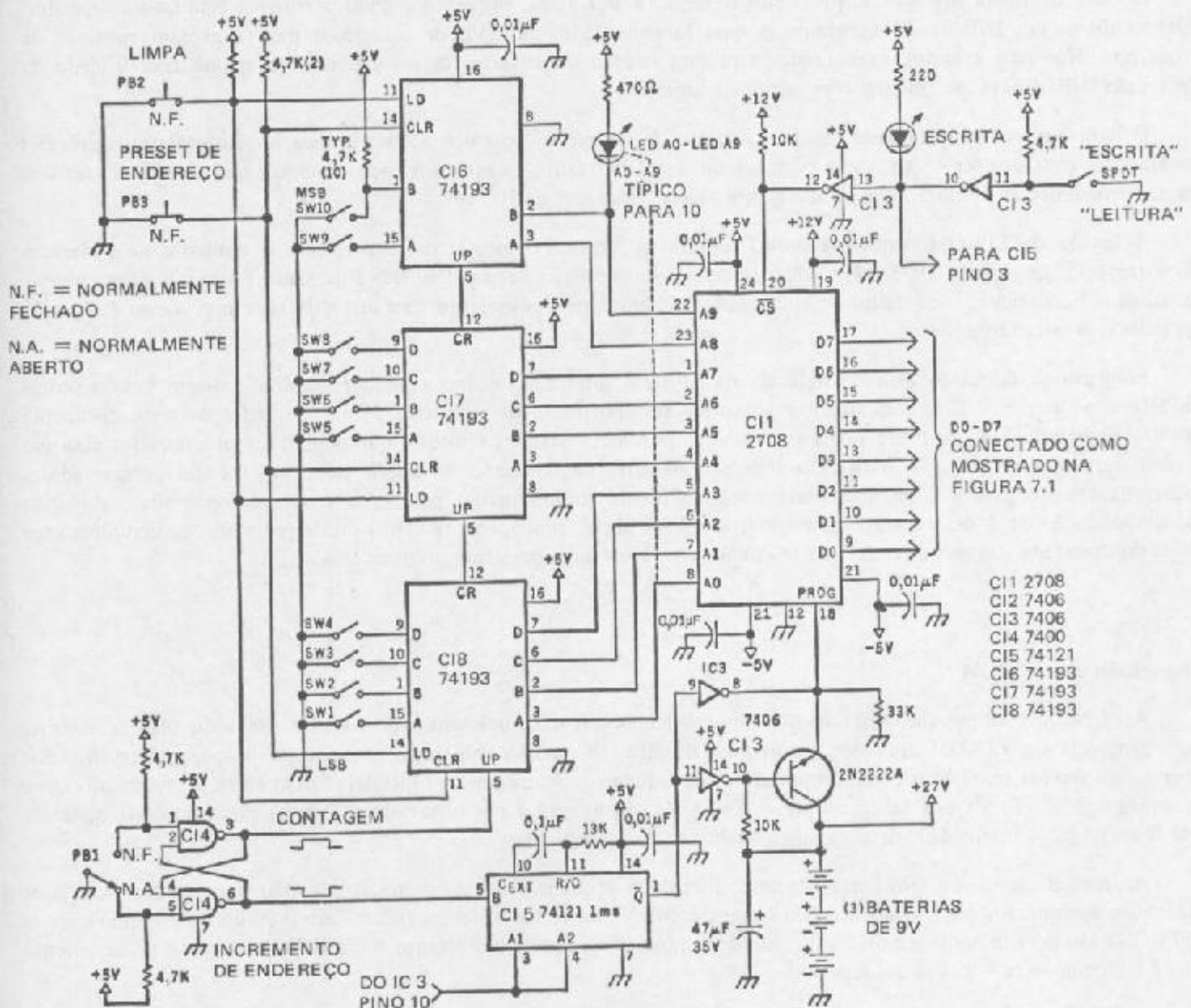



Figura 7.2 Diagrama esquemático de um programador manual auto-incrementado para 2708. Os diodos emissores de luz (LEDs) devem ser conectados em todas as 10 linhas de entrada de endereço da 2708. Somente um LED (conectado na linha de endereço A9) está mostrado no diagrama. Os outros LEDs devem ser colocados da mesma forma.

Quatro strobes de E/S (1 de entrada e 1 de saída para porta 1, e 1 de entrada e 1 de saída para porta 4) sincronizam o hardware e software. A figura 7.4 mostra o fluxo lógico para escrita de uma EPROM. Com a EPROM conectada diretamente à via de dados, somente os strobes, ao invés de registros de armazenamento completo, são necessários para esta interface.

Para escrever o dado, a seqüência deve ser: primeiro, um OUT 04 pulsa as linhas de clear dos contadores de endereço, colocando-os em 0, em seguida, a EPROM é colocada no modo de programação e o primeiro byte é escrito na EPROM com uma instrução de OUT 01.

A figura 7.5 mostra como o modo de programação da 2708 é selecionado. A importância deste circuito é que sua saída é ligada como um conversor de 2 bits digital por analógico para controlar a linha de chip-select da 2708.

Quando um OUT 04 é executado, o pino $\overline{CS}$ estará com 0V habilitando, assim, o modo leitura. Quando um OUT 01 é executado, esta tensão será de 12V para o modo de programação. Quando nenhum strobe estiver presente, $\overline{CS}$ estará em +5V e a 2708 estará em three-state (terceiro estado).

O OUT 01 libera o pulso de programação de 25V por 1 ms, enquanto o dado pertinente está na via de dados. Depois disto, um INP 01 é executado, o qual incrementa o contador de endereços para a próxima posição de endereço. Nós não estamos executando nenhuma função de entrada, estamos usando o strobe decodificado da instrução INP 01 para incrementar o registro de endereço.

O hardware automaticamente mantém o curso de endereço, porém o software deve implementar seus próprios contadores para manter o curso das posições de 0 a 1023 tanto quanto o número de vezes que os 1024 bytes têm de ser programados. Lembre-se que o fabricante sugere repetições de 100 vezes 1 ms.

A leitura da EPROM também é muito simples. A figura 7.6 mostra o fluxo lógico. O contador de endereços é novamente limpo pelo OUT 04. O dado é lido pela execução de um INP 04. Este dado pode ser armazenado e analisado. Finalmente, o contador de endereços é incrementado novamente com um INP 01, e o processo é repetido para ler o próximo byte.

Enquanto a discussão está centralizada na EPROM Intel 2708 como a melhor escolha, existem muitas outras EPROMs no mercado. Dois componentes de particular importância são os Intel 2758 e 2716. Estes são respectivamente EPROMs de 1K e 2K de fonte única (+5V). A importância destes circuitos é que podem ser programados com um único pulso de 50 ms, 25V para cada endereço ao invés de repetições sucessivas de 1 ms. Os três programadores apresentados são para a 2708, mas podem ser facilmente reconfigurados para estes outros componentes. Mudando a temporização de 1 ms para 50 ms e modificando-se alguns pinos, conseguiremos uma programação completa com uma única rodada através dos endereços (eles não têm de ser sucessivamente programados).

Apagando uma EPROM

As EPROMs compradas diretamente dos fabricantes vêm completamente apagadas. Se você planeja escrever um programa em EPROM uma vez, e não quer modificá-lo ou não cometer erros, esqueça o apagamento. A maior parte das pessoas que lidam com computadores irão querer reprogramar as EPROMs. Então torna-se necessário saber como apagá-las. Todos nós sabemos que as EPROMs são apagadas por ultravioleta. Entretanto, a duração, distância da fonte de luz, e intensidade determinam a qualidade do apagamento.

As especificações de fabricante seguidas durante a seqüência de programação são tão importantes quanto os métodos apropriados para apagamento. Diferentemente do teste "lê após ter escrito" do método de programação, as EPROMs são normalmente removidas do circuito durante o apagamento. Portanto, é aconselhável seguir o procedimento corretamente, ou este terá de ser repetido.

A EPROM 2708 pode ser apagada através da exposição à luz ultravioleta de onda curta de alta intensidade, com um comprimento de onda de 2537 Å. A dose recomendada (intensidade de UV × tempo de exposição) é de 12,5 watt-segundos por centímetro quadrado (Ws/cm^2). O tempo necessário para produzir esta exposição é uma função da intensidade da luz ultravioleta.

Custo e segurança, igualmente enfatizados, devem ser os fatores principais quando da seleção de um apagador de ultravioleta. Uma unidade comercial não só especifica sua intensidade, mas também inclui especificações de segurança.

Tempo de exposição (T_E)

$$T_E = J \div I$$

onde

J = densidade de apagamento requerida pelo componente
I = densidade de potência incidente do apagador

Para a 2708 é necessário 12,5 Ws/cm^2

$$I = 5000\ \mu W/cm^2$$
$$J = 12{,}5\ Ws/cm^2$$

$$T_E = \frac{12{,}5}{5000 \times 10^{-6}} = 2500 \text{ segundos}$$

ou $T_E = 41{,}6$ minutos

Um dos melhores apagadores no mercado é o UVS-11E fabricado pela ULTRA-VIOLET PRODUCTS, INC, SAN GABRIEL CA, 91776. Esta unidade é feita especialmente para o mercado de computadores pessoais e inclui algumas facilidades importantes de segurança. A lâmpada não irá acender se não estiver apoiada e se for levantada do seu local de apoio. Na distância padrão de uma polegada, o USV-11E produz uma intensidade de 5000 μW por centímetro quadrado. O tempo de exposição para a 2708 pode, então, ser facilmente calculado.

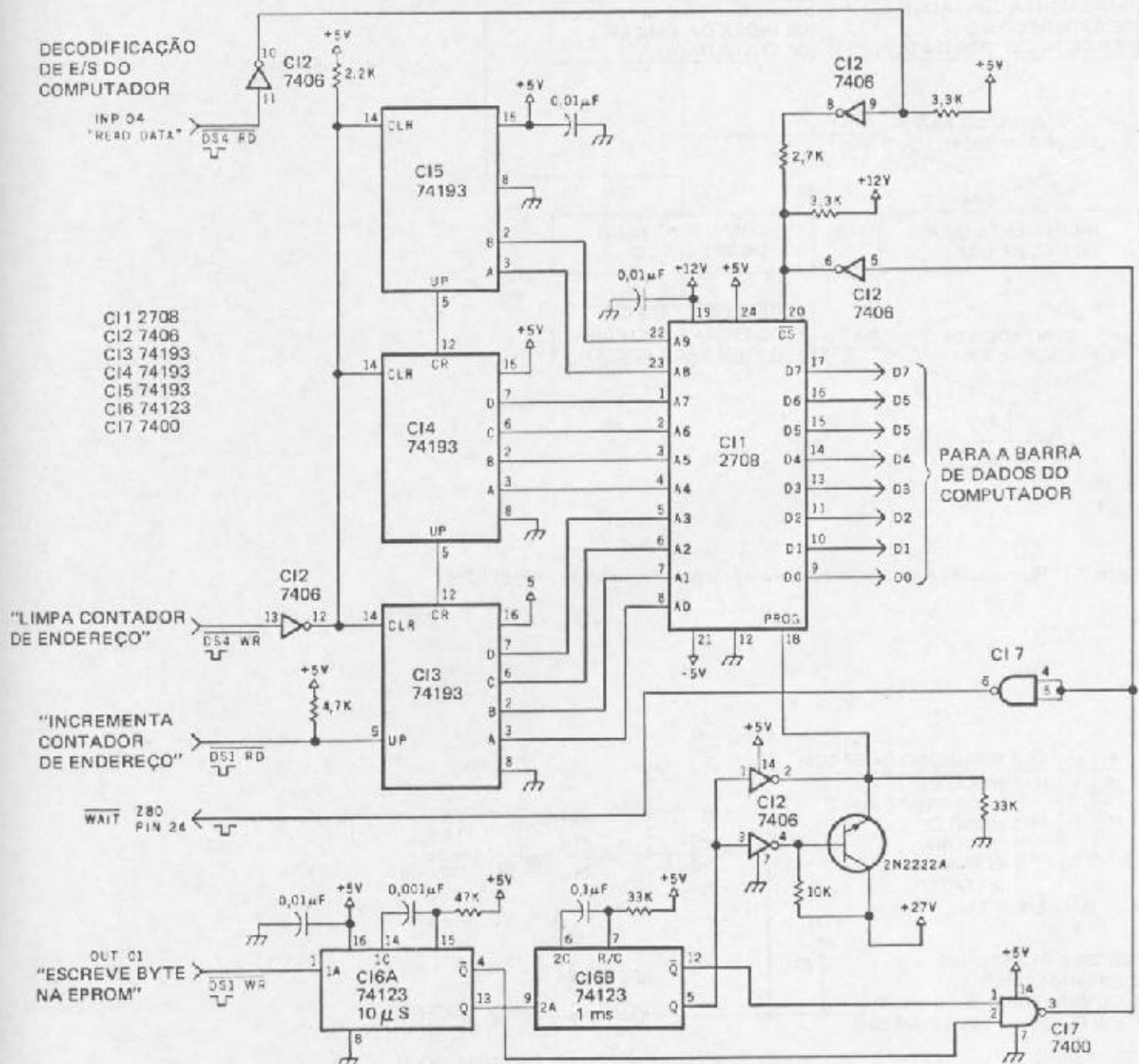



Figura 7.3 Diagrama esquemático de um programador automático para 2708.

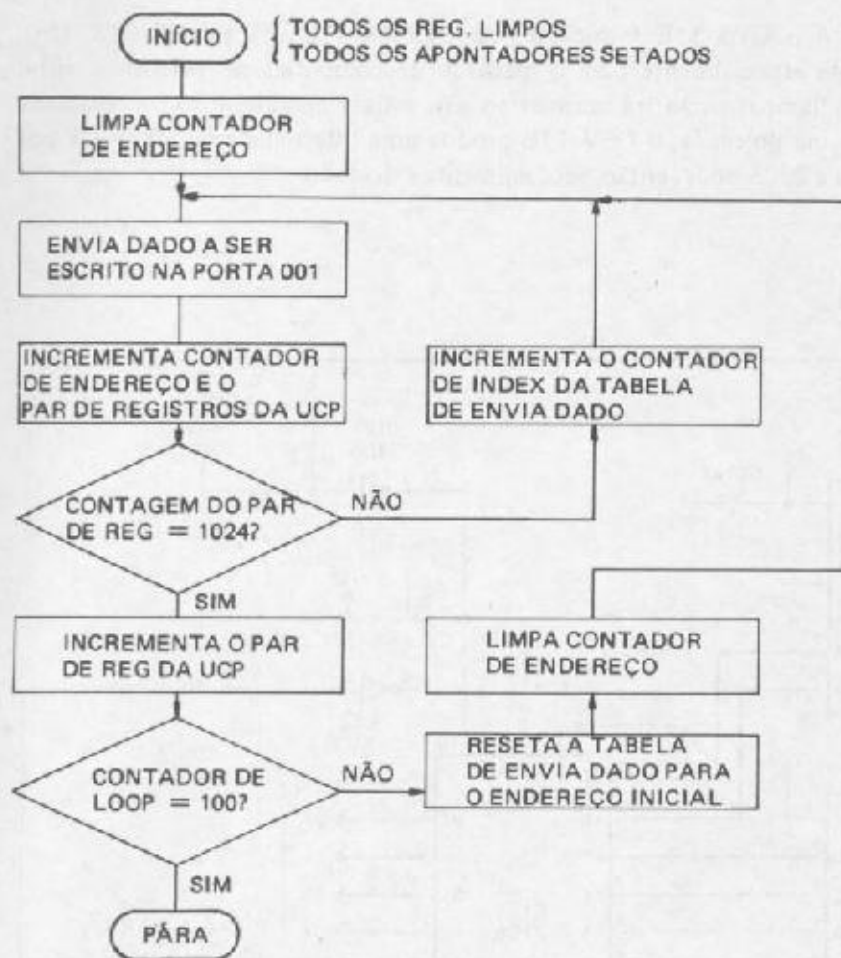



Figura 7.4 Fluxograma do ciclo de escrita de um programador automático de EPROM.

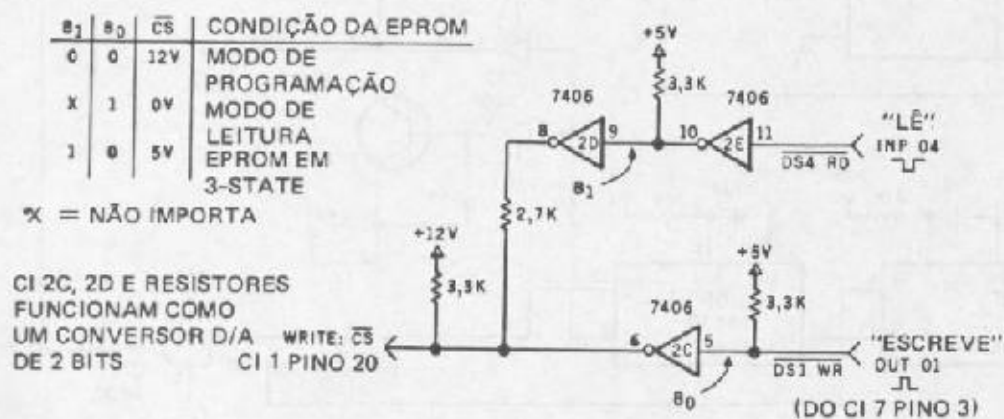



Figura 7.5 Controle programável da linha de seleção de uma EPROM $(\overline{CS})$ de um programador automático.

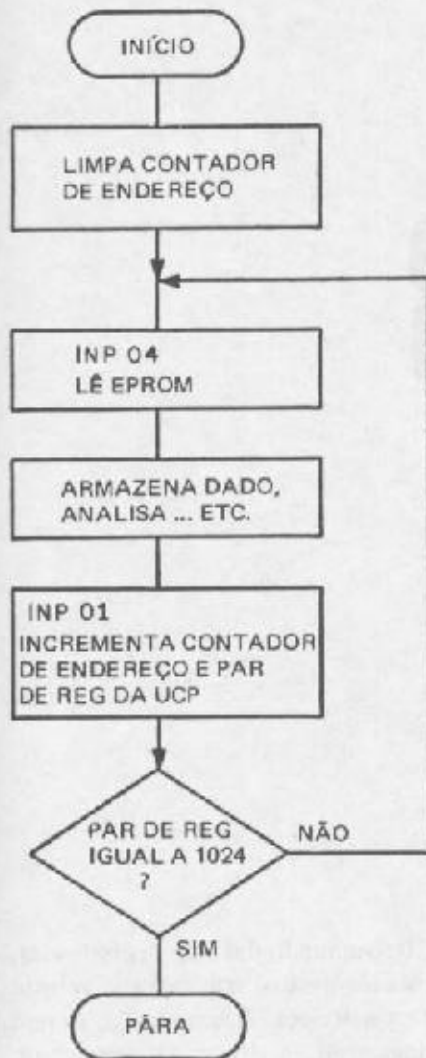



Figura 7.6 Fluxograma de um ciclo de leitura de um programador automático de EPROM.

CAPÍTULO 8

CONECTANDO O PAZ COM O EXTERIOR

É óbvio agora que o computador PAZ pode ser configurado de muitas maneiras. Dependendo das suas necessidades, você pode ir além do sistema básico que foi apresentado. Se você quiser um computador pessoal que seja equivalente a um microcomputador comercial, então você deve adicionar bem mais dinheiro e periféricos. Acomodações devem ser feitas para um sistema operacional mais eficaz e muito provavelmente uma linguagem de alto nível assim como BASIC ou PASCAL. Se você pretende usar o computador PAZ como um sistema de processamento da palavra, então precisa de um terminal vídeo e uma impressora. Para isso é necessário o acréscimo de portas seriais e paralelas. Qualquer que seja a configuração, as considerações de projeto que foram vistas na construção do computador PAZ não mudam.

O computador PAZ é voltado para a parte de treinamento. Este livro é estruturado de forma que você será capaz de desenhar uma configuração de sistema e montá-la.

Eu não demonstrei como se faz para projetar um sistema de processamento da palavra ou como adicionar um FLOPPY, porque está além dos objetivos deste texto. O material para cobrir estes projetos necessitaria de um outro livro. Não significa, entretanto que tudo está terminado, uma vez que o PAZ esteja construído e você tenha aprendido como programá-lo. Muito pelo contrário, uma aplicação mais significativa do PAZ é ligá-lo à alguma coisa considerada do mundo real e fazê-lo executar tarefas. A chave real para usar o PAZ é realmente ligá-lo ao mundo real.

Desde o início eu citei o computador PAZ como sendo um computador em uma única placa que pode ser usado em uma grande variedade de aplicações. Como ele inclui uma porta serial, duas portas paralelas, um monitor em PROM, e memória programável, o PAZ em muitos aspectos se equivale a um controlador digital comercial que custa uns milhares de cruzeiros a mais. Pequenos computadores são geralmente mais usados em aquisição de dados e aplicações de controle. Suas funções são geralmente digerir certos parâmetros de entrada e computar um resultado.

Por exemplo, em um controle de um motor elétrico de 100 HP, as entradas seriam voltagem, corrente e RPM, e o controle de saída seria um fator de correção de tensão de carga.

Em muitos casos, as funções não se limitam a controles simples. Em algum processo onde repetição e controle de qualidade são importantes, parâmetros do processo são constantemente monitorados para limitar o desvio de limites pré-estipulados, e um alarme é ligado se estes limites são excedidos. Para ajudar na função de aquisição de dados, geralmente é incluído o armazenamento de dados vindos de sensores a intervalos específicos para que um registro permanente seja gerado.

O mundo real

Eu não quero confundi-lo discutindo sobre tantas aplicações comerciais.

Existem muitas aplicações caseiras como controle de energia, segurança e monitoração do ambiente. Eu me refiro a estes sistemas como sistemas do mundo real.

Como o mundo real é alguma coisa fora do computador, geralmente é um mundo analógico e não digital.

A metamorfose do PAZ em um controlador inteligente é dependente primeiramente de uma interface analógica. Por esta razão o resto deste capítulo é dedicado ao projeto e construção de uma interface de E/S analógica que seja econômica.

Mas primeiro vamos rever alguns pontos básicos da conversão analógico-digital.

Conversores digitais-analógicos

O conversor D/A pode ser imaginado como um potenciômetro controlado digitalmente para produzir uma saída analógica. Este valor de saída ($V_{saída}$) é o produto de um sinal digital (D) e uma referência analógica (V_{REF}) e é expresso pela seguinte equação:

$$V_{SAÍDA} = D\, V_{REF}$$

De uma forma geral, nenhum conversor é muito útil sem que se especifique o tipo de código usado para representar a magnitude digital. Os conversores trabalham ou com código digital unipolar ou bipolar. Unipolar inclui binário e binário codificado em decimal (BCD). Bipolar inclui complemento a um e a dois, e o código GRAY.

É importante lembrar que uma quantidade binária apresentada pelo computador é a representação de um valor fracionário que será multiplicado por uma tensão de referência. Em frações binárias o bit mais significativo tem o valor de 1/2 ou 2^{-1} e o próximo de 1/4 ou 2^{-2}, e o menos significativo $1/2^n$ ou 2^{-n}, onde n é o número de dígitos binários depois da vírgula. Somando-se todos os bits produz-se um valor que se aproxima de 1. Quanto mais bits houver, mais se aproxima de 1. A diferença algébrica entre o valor binário que se aproxima de 1, e 1, é o erro quantitativo do sistema digital.

A conversão de valores digitais para valores analógicos proporcionais é conseguida através de dois tipos de conversão: o conversor de resistores com valores específicos (WEIGHTED-RESISTOR) e o conversor R-2R.

O conversor de resistores com peso específico é o mais simples e o mais fácil. Este decodificador paralelo requer um resistor por bit e funciona da seguinte maneira: as correntes com valores de 1/2, 1/4, 1/8, ..., $1/2^n$ são gerados por resistores com valores de R, 2R, 4R, ..., 2^nR, que são conectados por chaves entre a tensão de referência (V_{REF}) e o ponto de soma de um amplificador operacional. As várias correntes são somadas e convertidas em voltagem por um OP AMP (veja figura 8.1).

Enquanto isto pode parecer uma resposta simples para um problema complexo, este método tem alguns problemas. A precisão deste conversor é uma função das precisões combinadas dos resistores, chaves (todas as chaves apresentam alguma resistência) e do amplificador de saída.

Em sistemas de conversão maiores do que 10 bits de resolução, os valores dos resistores tornam-se excepcionalmente grandes e o fluxo de corrente é reduzido a um valor tão pequeno que chega a ser confundido com o ruído térmico do sistema.

Uma alternativa razoável é o uso do conversor R-2R. Este conversor é o mais usado, embora use mais componentes. O circuito da figura 8.2 tem também uma tensão de referência, um conjunto de chaves binárias e um amplificador de saída. A base deste conversor é uma malha em escada construída com 2 resistores, R e 2R.

Um resistor (2R) está em série com a chave de bit, enquanto o outro (R) está na linha de soma, esta combinação forma uma malha em π.

Isto faz com que a impedância das três ramificações de qualquer nó seja igual, e que uma corrente I fluindo em um nó através de uma ramificação passe para as outras ramificações com um valor de I/2. Em outras palavras, uma corrente produzida ao se fechar a chave de bit é reduzida à metade na sua passagem em cada nó até o fim da escada. A posição da chave com relação ao ponto onde a corrente é medida é que determina o valor binário desta chave em particular.

$$V_{SAÍDA} = -\left[\frac{-V\,REF}{R} + \frac{-V\,REF}{2R} + \frac{-V\,REF}{4R} + \frac{-V\,REF}{8R}\right] RF$$

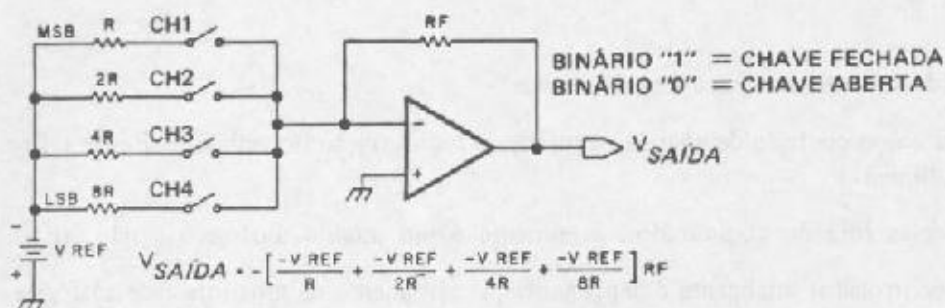



Figura 8.1 Conversor analógico-digital do tipo Weighted-resistor

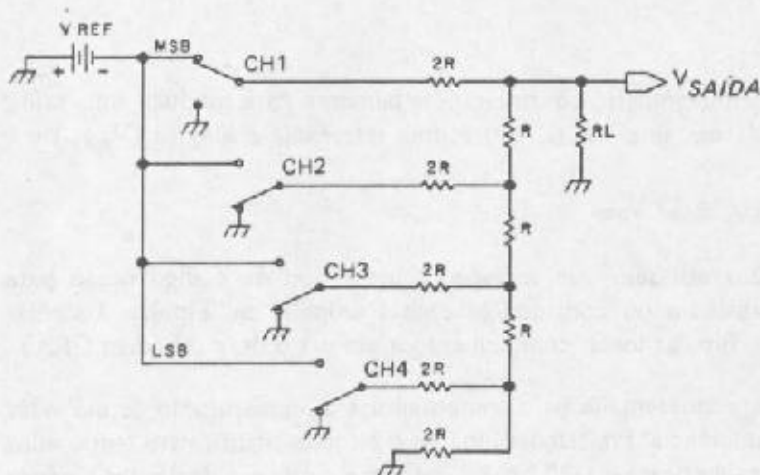



Figura 8.2 Conversor analógico-digital do tipo R-2R.

Este tipo de conversor é fácil de ser fabricado porque só dois valores de resistores são usados. Se mantivermos os valores dos resistores os mais próximos possível e com o mesmo coeficiente de temperatura isto contribuirá para um projeto bem estável. Certos requisitos são necessários entre os valores dos resistores da escada e o fluxo da corrente para contrabalançar a precisão e o ruído.

Uma forma de circuito R-2R é o conversor D/A de multiplicação e é conseguido com uma referência fixa ou uma referência externa variável. Conversores D/A de multiplicação que utilizam tensão de referência externa variável produzem saídas que são diretamente proporcionais ao produto da entrada digital multiplicada por esta referência variável. Esses componentes têm saída tanto em tensão como em corrente. Os componentes com saída em corrente são muito mais rápidos, porque eles não têm amplificador de saída que limita a largura de faixa, além do que eles tendem a ser mais baratos do que o de saída em tensão. Um exemplo é o conversor D/A da MOTOROLA MC1408-8 (veja figura 8.3).

Cada bit controla uma chave que regula a corrente que flui na escada. Se uma entrada digital de 8 bits de 11000000 fosse aplicada às linhas de controle, a corrente de saída seria igual à (192/256) X (2 mA) ou 1,50 mA. Note que quando o número binário 11111111 for aplicado, a máxima corrente de saída será de 1,92 mA para uma corrente de referência de 2 mA. Esta diferença é que determina a precisão do conversor, que no caso é de 0,19% da máxima escala (veja figura 8.4).

A figura 8.5 mostra o circuito final de um conversor de 8 bits. A tensão de referência de 6,8V dada por um diodo-zener passa através de um resistor que irá fornecer a corrente de aproximadamente 2 mA ao pino 14 do integrado.

Um resistor adicional R1 permite que a corrente seja variada de um pequeno valor para permitir ajuste da máxima escala do conversor. A saída é uma corrente que é equivalente ao produto desta corrente de referência e do dado binário nas linhas de controle. A corrente é convertida em tensão através de CI9 e pode ser ajustada a zero pelo potenciômetro R2.

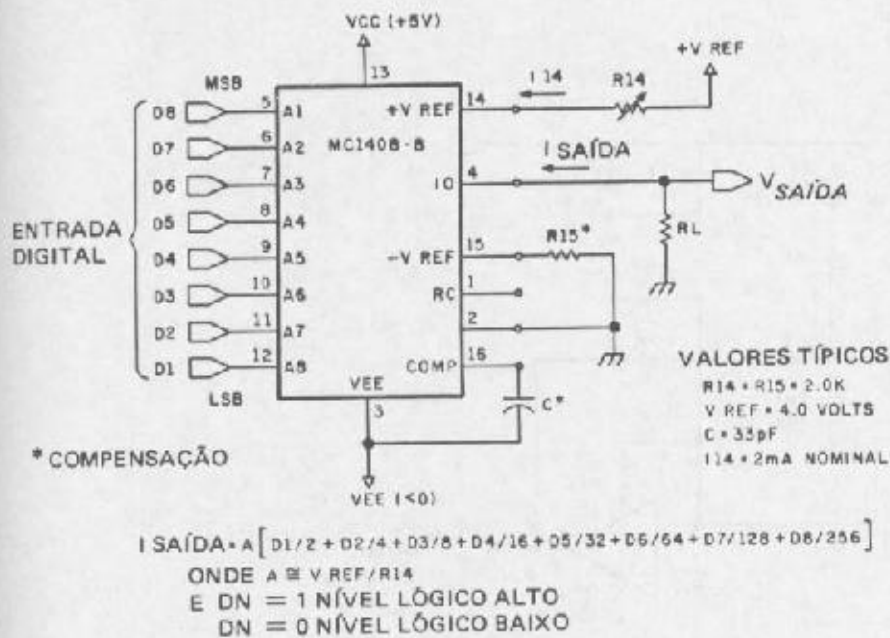



Figura 8.3 Conversor D/A de 8 bits do tipo R–2R.

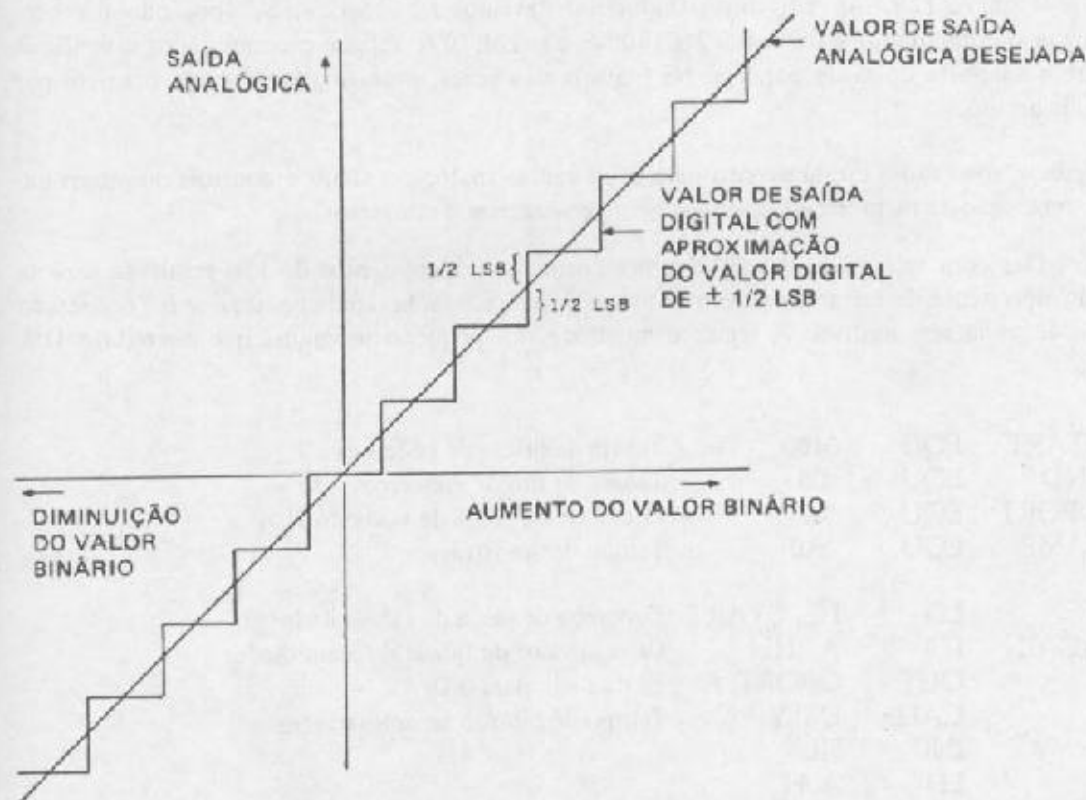



Figura 8.4 Características de saída de um conversor D/A típico.

Para usar este circuito com o PAZ basta ligar as linhas de entrada do CI1 a uma porta de saída do PAZ. Qualquer valor de 8 bits enviado a esta porta será convertido em uma tensão proporcional à saída.

A calibragem é fácil de ser feita. Ligue o computador, e com um pequeno programa que envia um valor do acumulador, envie o número binário 10000000 à porta de saída correspondente ao endereço da interface D/A. Usando um medidor para monitorar a saída do LM301A ajuste o potenciômetro de zero R2 até obter 0V na

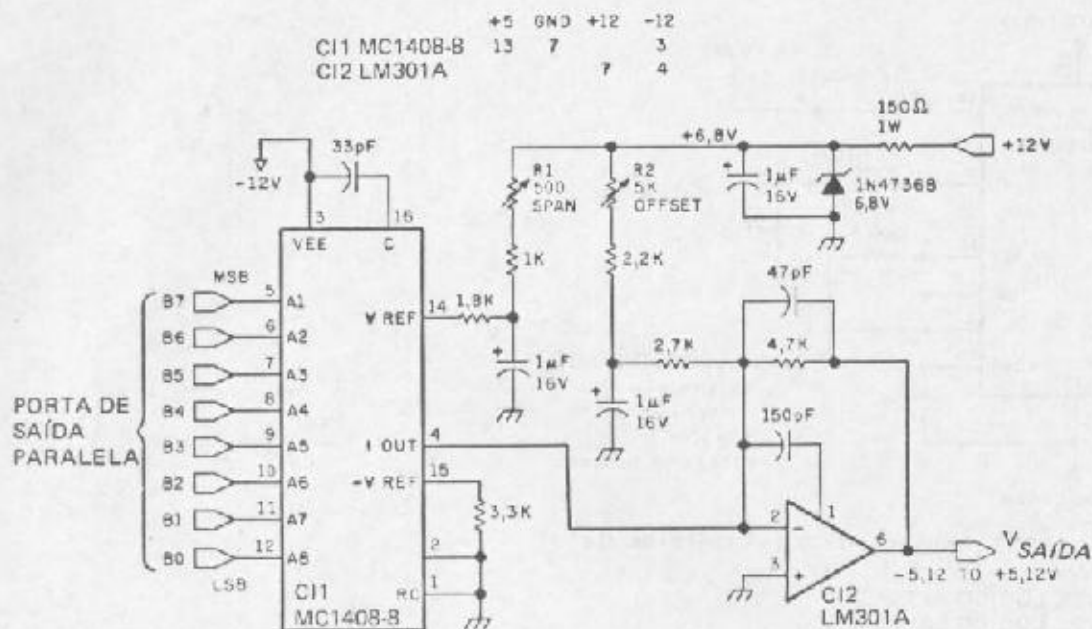



Figura 8.5 Conversor final D/A de 8 bits usando o integrado MC1408-8.

saída. Com o mesmo programa carregue o número 11111111, envie à porta de saída e ajuste o potenciômetro R1 para uma leitura no medidor de +5,12V. Ao enviarmos 00000000 devemos ler –5,12V. Se você não for bem sucedido neste ponto, desligue o computador e remova o MC1408-8 e o LM301A, religue o computador e verifique se a saída binária está correta na porta de saída paralela. Na maioria das vezes, problemas como este ocorrem por se escolher um código de saída errado.

Se o teste for bem sucedido, você agora estará pronto para gerar saídas analógicas sobre o controle do programa. Um teste simples é designar uma seção da memória e seqüencialmente enviar para o conversor.

Se a tabela for de 256 bytes com valores de 0 a FF em hexadecimal em incrementos de 1, o resultado será na saída uma forma de onda do tipo dente de serra. Se os valores forem enviados à saída rapidamente, e se for conectado a um alto-falante, a forma de onda será audível. A seguir é mostrado um pequeno programa que exercita o D/A da maneira explicada acima.

```
START   EQU   0400        Tabela de início de endereço
END     EQU    05         Tabela de fim de endereço
OPORT   EQU    07         Endereço da porta de saída do D/A
SAMP    EQU    A0         Tempo de amostragem

        LD    HL, START   Endereço de carga da tabela de início
AGAIN   LD    A, (HL)     Carrega valor da tabela no acumulador
        OUT   OPORT, A    Envia dado para o D/A
        CALL  DELY        Tempo de retardo na amostragem
        INC   HL
        LD    A, H
        CP    END         Testa se fim de tabela
        JP    NZ, AGAIN   Senão, envia próxima amostragem
        HALT

DELY    LD    B, SAMP     Amostra razão de temporização do loop
DCR     DEC   B
        JP    NZ, DCR
        RET
```

A tabela pode ser colocada para qualquer tamanho. Os valores na tabela podem ser calculados para produzir qualquer forma de onda.

Conversores analógico/digital

Um conversor A/D faz jus ao seu nome. Ele converte voltagens analógicas em binário. Como no caso de um D/A de 8 bits, um A/D está sujeito às mesmas leis de conversão. Se você tentar ler um sinal de 10V com um de 8 bits, a resolução será de 1/256 de 10V (ou 40 mV) e a precisão será de ± 1/2 bit menos significativo.

Para resoluções maiores mais bits são necessários. Um conversor de 8 bits pode ser facilmente ajustado para cobrir uma área de 0 a 1V ou de 0 a 1000V. Geralmente o mesmo circuito é usado, exceto o estágio final de amplificação e a malha de resistores que são trocados.

Para o computador PAZ a questão sobre qual conversor usar e com que precisão é mais uma questão de preço.

A conversão analógica-digital é consideravelmente mais cara do que D/A, o preço está diretamente ligado à resolução e à precisão. Existem várias maneiras de se fazer a conversão A/D. Um conversor A/D pode custar alguns cruzeiros bem como milhares de cruzeiros.

Por este motivo foram escolhidos quatro tipos de conversores, espero que algum deles satisfaça as suas necessidades.

1. Conversor analógico para largura de pulso básico.
2. Conversor contador de rampa de 8 bits de baixo custo e baixa velocidade.
3. Conversor de aproximação sucessiva de 8 bits de alta velocidade.
4. Interface de oito canais de 3 1/2 dígito de 0–200V CA/CC.

CONVERSORES DE LARGURA DE PULSO E CONTADORES

Conversor analógico para largura de pulso

Este conversor é um dos mais populares codificadores de elo-aberto, devido à sua simplicidade. Um diagrama de bloco básico é mostrado na figura 8.6. Este elemento usa um oscilador fixo em combinação com um circuito que gera um pulso que é uma função linear da tensão analógica da entrada.

Para se obter este pulso linear variável, os projetistas geralmente usam um gerador de rampa e um circuito Schmitt-trigger. Um pulso é iniciado no começo da rampa e um circuito contador começa a incrementar em uma frequência fixa. Quando a rampa linear chega ao mesmo valor da tensão de entrada, a contagem termina. O valor deixado no registrador neste ponto é o equivalente à entrada analógica.

A figura 8.7 mostra o esquema de um conversor que opera neste princípio. O CI1 é configurado como um gerador linear de rampa controlado e o CI2 é o comparador da entrada. O processo inicia-se quando o clock de 7,5 KH dispar o CI3 (monoestável 74121) e começa o seu período de 35 ms, que é o tempo de disparo. No início deste tempo de disparo, é gerado um pulso que limpa dois 7493 e o gerador de rampa é zerado.

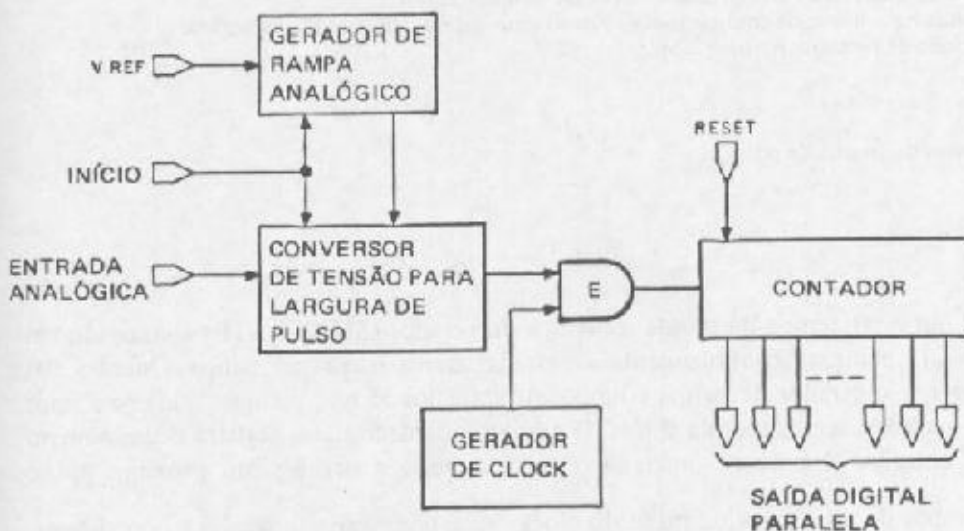



Figura 8.6 Diagrama de blocos de um codificador de largura de pulso.

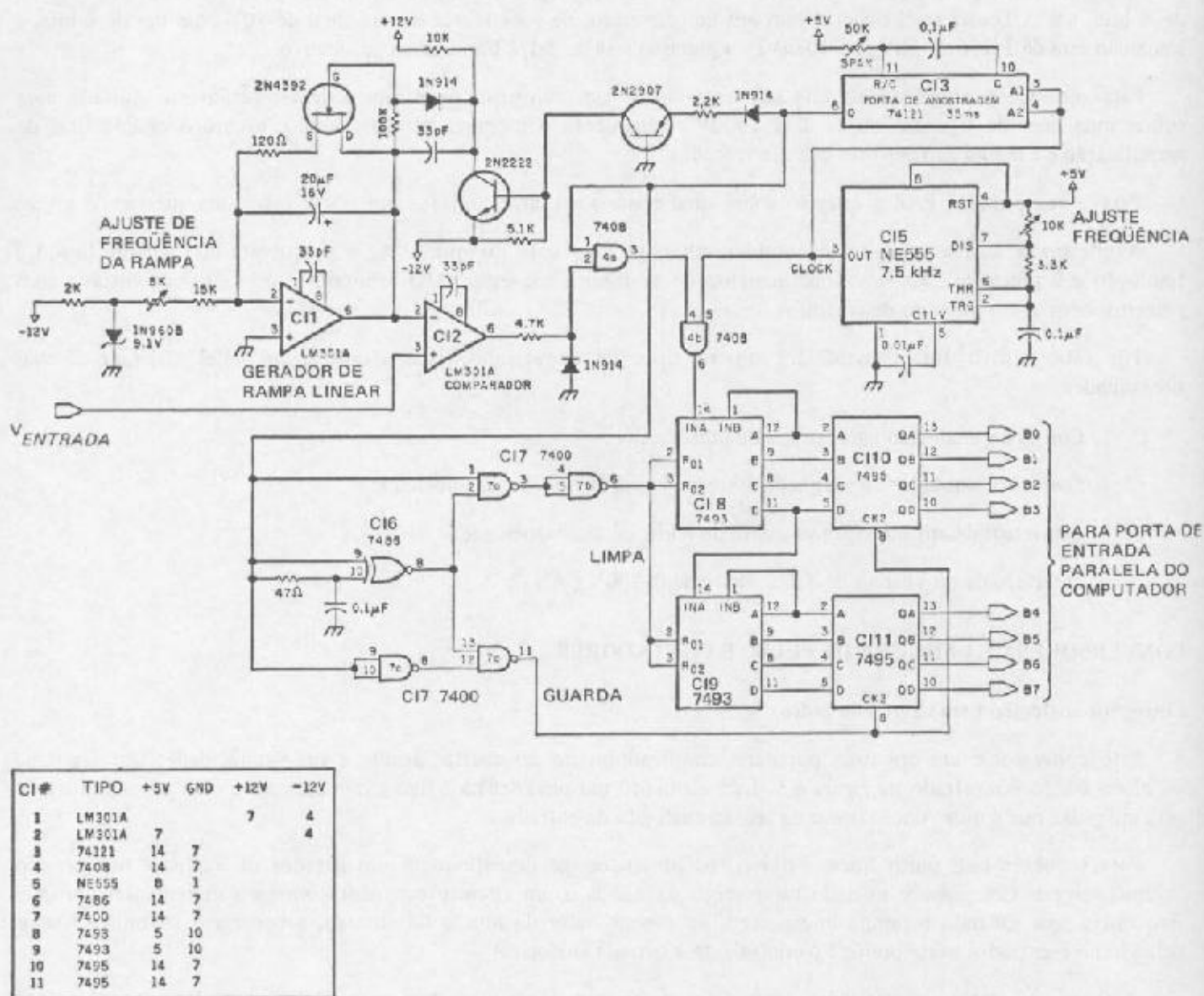
CI#	TIPO	+5V	GND	+12V	-12V
1	LM301A			7	4
2	LM301A	7			4
3	74121	14	7		
4	7408	14	7		
5	NE555	8	1		
6	7486	14	7		
7	7400	14	7		
8	7493	5	10		
9	7493	5	10		
10	7495	14	7		
11	7495	14	7		

NOTAS: 1. Faça com que a rampa vá de 0V ao máximo da escala durante o tempo de amostragem.
2. Coloque a freqüência para produzir o número de contagens que se queira que represente a tensão de entrada. Ex.: contar 256 durante o período de amostragem para 2,56V.

Figura 8.7 Diagrama esquemático do conversor de largura de pulso.

Isto faz com que a contagem se inicie. O tempo de subida da rampa do gerador (SLEW RATE) é colocado em 10V em 35 ms aproximadamente. O CI2 compara continuamente a entrada com a tensão da rampa. Quando elas são iguais, o clock pára, o contador pára, e o gerador de rampa é limpo. Ao final dos 35 ms, qualquer que seja o valor que esteja no contador, é transferido para um registrador de 8 bits. O número guardado neste registro é um número de 8 bits proporcional à tensão de entrada. O processo inicia-se outra vez quando ocorrer um próximo pulso.

Ao selecionar propriamente os tempos de abertura e da razão do clock, você pode variar a resolução do sistema. Com um tempo de abertura de 35 ms e um clock de 7500 Hz, 256 clocks devem ser contados durante o tempo de abertura.

O potenciômetro de ajuste do tempo da rampa deve ser colocado para que o contador chegue ao máximo quando 2,56V for aplicado à entrada do CI2. Um divisor de 10:1 acoplado a esta entrada permitirá que o mesmo contador de 8 bits represente 25,6V.

Este circuito é simples, mas sua precisão depende da estabilidade dos diversos circuitos individualmente.

Para usá-lo coloque a saída do registrador de saída na porta paralela. Simplesmente leia a porta quando você quiser o último valor. O circuito se atualiza automaticamente 28 vezes cada segundo, por isso nenhuma letra é mais antiga do que 35 ms.

Conversor contador de rampa

A técnica de A/D acima é mais usada em grandes períodos de amostragem e alta precisão nas medidas. Para se chegar a estes resultados deve-se usar, entretanto, componentes de precisão e ter-se uma montagem própria.

O próximo circuito a se discutir é o método do contador de rampa. Na minha opinião este é o melhor tipo, se você está pensando em construir um A/D para o PAZ.

Ele usa poucos componentes e, na prática, é mais rápido e fácil de se construir do que os circuitos lineares de rampa.

A figura 8.8 mostra o diagrama de blocos básico para o conversor contador de rampa binário. O gerador de rampa linear da técnica descrita anteriormente foi substituído por um conversor D/A. Neste caso D/A é usado para reconverter a saída digital do contador binário, de volta a um valor analógico para comparação com a entrada analógica. Se elas forem iguais então o conteúdo do contador será o valor convertido que queremos.

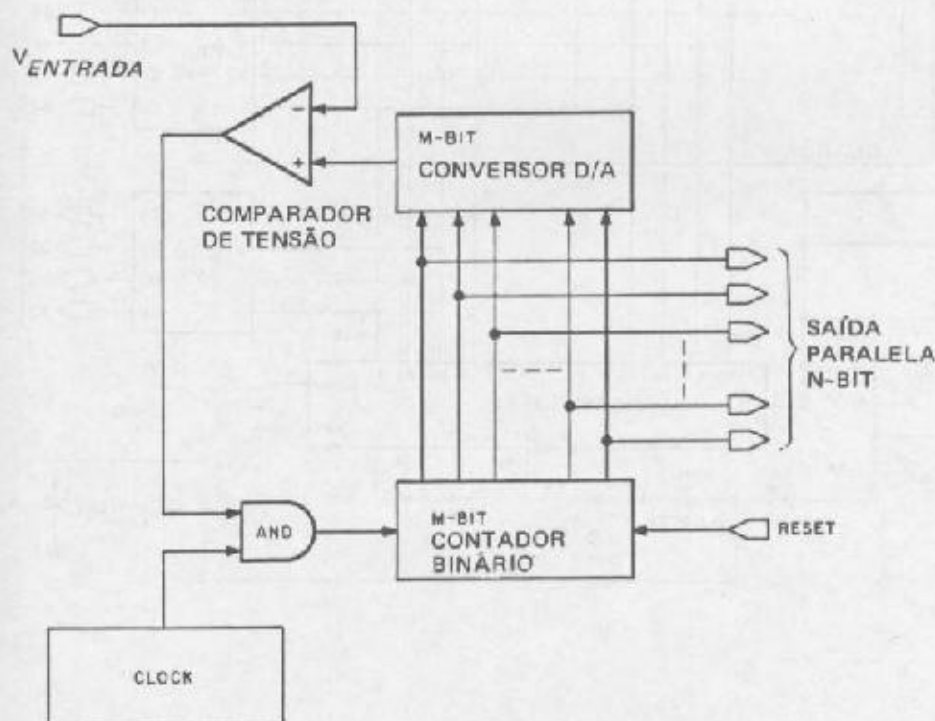



Figura 8.8 Diagrama em blocos de um conversor A/D do tipo contador de rampa binário.

A maneira mais simples de se operar o sistema é começar o contador com 0 e permitir que ele conte até que o D/A se iguale ou exceda à entrada analógica. A única consideração crítica no projeto destes circuitos é que a freqüência do clock não pode ser mais rápida do que a resposta do comparador ou do D/A. Se levar 100 μs para estes componentes fazerem o seu trabalho, então a máxima razão de clock deve ser de 10 KHz.

Para um conversor de 8 bits (contando de 0 a 256 a cada período de amostragem), a razão máxima de amostragem é 10.000/256 ou 39 amostragens por segundo. Na prática entretanto 5μs é um tempo razoável, o que equivale a 750 amostragens por segundo. Para velocidades ainda maiores pode-se usar um tipo diferente de A/D, que veremos depois.

A figura 8.9 mostra o esquema de um conversor do tipo contador de rampa binário, que usa o integrado MC1408-8. A saída do contador está ligada ao MC1408-8 para termos uma comparação analógica direta do valor contido no contador.

Inicialmente os CI's 4 e 5 são limpos e a saída do D/A deve ser igual a menor voltagem. Para um conversor de 0 a 5,12V deve ser 0V. Para um de −2,56 a +2,56V deve ser de −2,56V. Se a saída do CI1 for menor do que $V_{ENTRADA}$ do comparador, os pulsos de clock estão permitidos de chegar ao contador. Como a cada pulso o contador é incrementado, a saída do D/A continua subindo até que seja igual ou exceda $V_{ENTRADA}$ no comparador. Quando isto acontece, os pulsos de clock são inibidos. No final do período de amostragem o valor dos contadores CI 4 e 5 é guardado em um registrador separado.

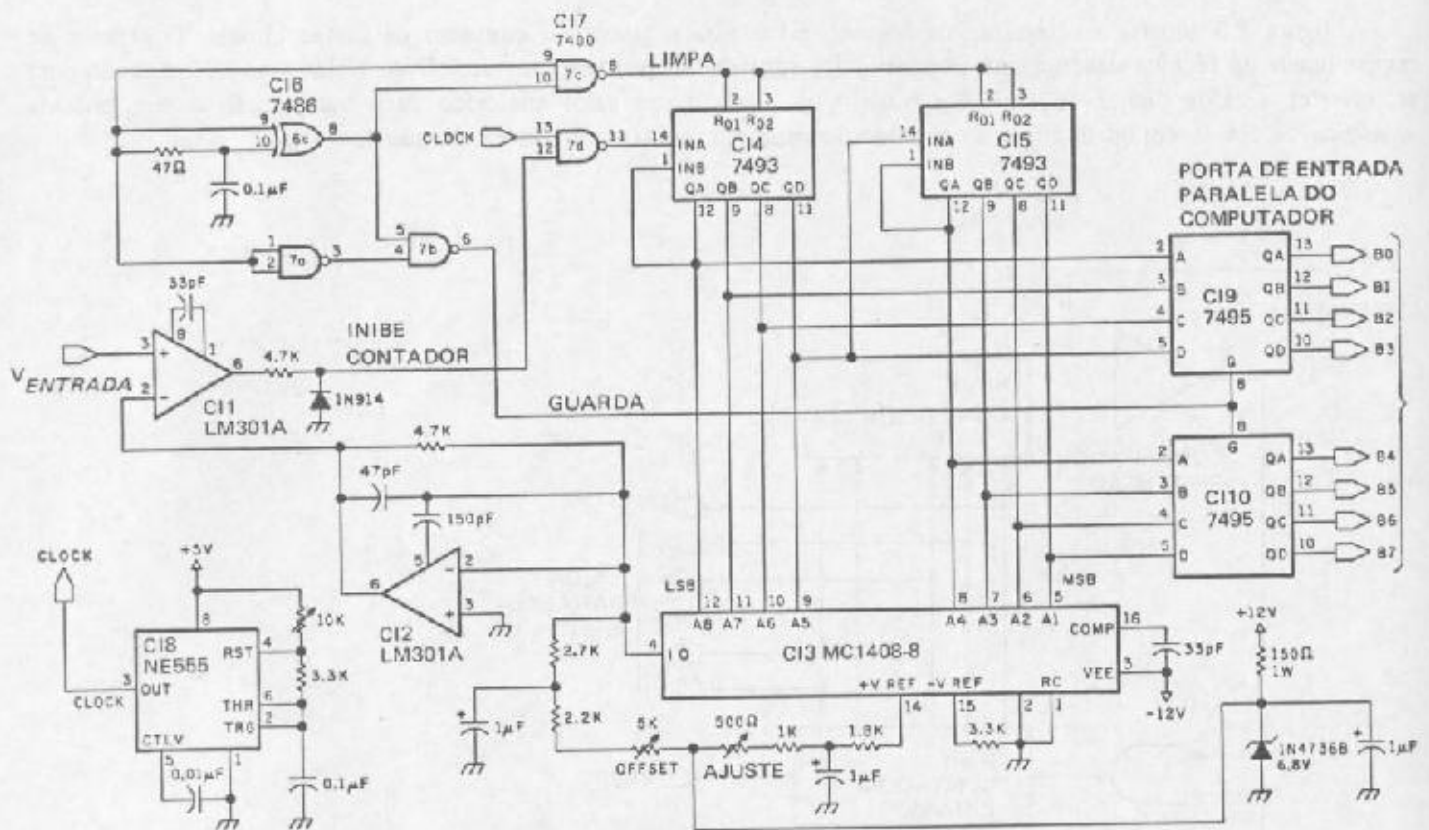
Para o PAZ ler este registro basta ligá-lo a uma porta de entrada e lê-lo diretamente.

CI #	TIPO	+5V	GND	+12V	−12V
1	LM301A	7			4
2	LM301A			7	4
3	MC1408-8	13	7		3
4	7493	5	10		
5	7493	5	10		
6	7486	14	7		
7	7400	14	7		
8	NE555	8	1		
9	7495	14	7		
10	7495	14	7		

Figura 8.9 Diagrama esquemático de um conversor A/D de 8 bits do tipo contador de rampa binário.

Usando o computador para substituir o contador

A figura 8.9 mostra um circuito que não precisa do computador para sua operação. O A/D atualiza a si mesmo a uma razão de amostragem predeterminada e carrega o seu valor em um registro de 8 bits.

Existem algumas vantagens com relação a este circuito. O A/D pode ser totalmente montado e testado sem um computador. Por exemplo, uma tensão pode ser aplicada à entrada e os 8 bits podem ser mostrados em LEDs.

Considere por um momento os elementos principais deste projeto. Este conversor A/D tem quatro seções: comparador analógico, D/A, contador de 8 bits e lógica de temporização. A escada de resistores e o comparador analógico são componentes necessários, mas as últimas duas seções são fortes candidatas a serem sintetizadas pelo computador. As funções destes elementos são incrementar um contador de 8 bits e testar a saída do comparador. O computador PAZ tem portas de entrada e saída paralelas. Incrementando-se um dos registros do processador central e enviando-se este valor após cada incremento, as 8 linhas da porta de saída parecerão como um contador. Usando-se um bit de uma porta de entrada para ler o *status* do comparador, nós podemos também substituir o resto da lógica de temporização.

A interface resultante passa a ter menos componentes e é mostrada na figura 8.10.

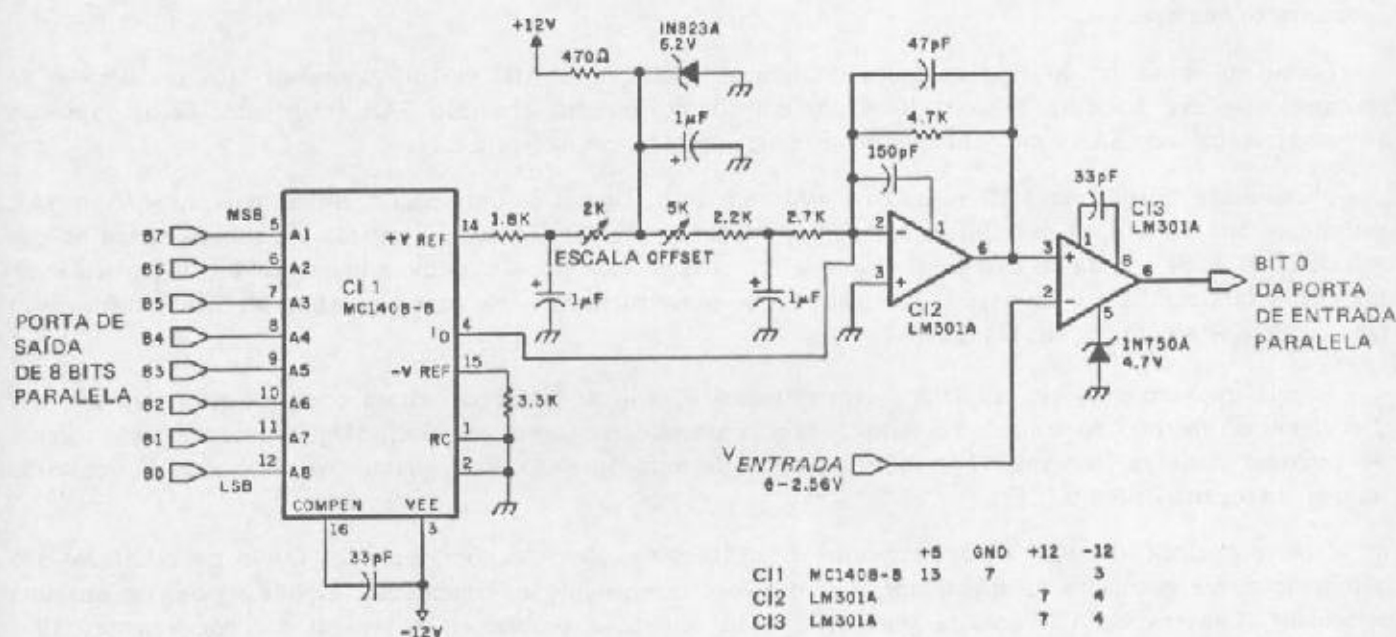



Figura 8.10 Conversor A/D de 8 bits comandado por software.

O processo de conversão não é diferente da outra versão feita só por componentes. Primeiro, nós limpamos um registrador do Z80 (por exemplo, o B) e então enviamos o valor deste registro para a porta que está ligada ao D/A.

Isto fará com que o D/A fique com sua saída mínima. A seguir, nós lemos a porta de entrada que tem o comparador ligado e testamos o bit 0 (um nível 1 indica que a voltagem de entrada e do D/A são iguais). Se o comparador estiver baixo (as tensões não são iguais), o registrador será incrementado e o processo será repetido. Quando a saída do comparador for 1, o valor convertido da tensão de entrada estará no registrador B, então o programa será parado. O programa que executa esta função é mostrado a seguir:

```
        MVI   B       limpa registro B
        OUT   0,B     envia registro B
AGAIN   INC   B       incrementa B
        OUT   0,B     envia registro B
        IN    04      lê comparador
        ANA   01      isola bit 0
        JNZ   AGAIN   repita se as tensões não forem iguais
        HLT           o valor da conversão está em B
```

O programa acima deve ser repetido a cada vez que uma nova leitura é necessária e a razão de amostragem pode ser ajustada dentro de certos limites. Lembre-se que nós ainda temos de esperar que o conversor D/A se inicialize e este não deve ser incrementado mais rápido do que 5 μs. O uso de um Z80 de 2,5 MHz não deve apresentar problema. O uso de um Z80 de 4 MHz necessitará de alguns NOPs no programa.

Existem muitas variações deste circuito. Como descrito, ele necessita de 255 interações do programa para achar uma resposta.

Em um computador com um tempo médio de instrução de 2 μs o programa levaria 3 μs para acabar, o que daria umas 300 amostragens por segundo. Some a isto as outras tarefas que o computador tem de fazer e chega-se a umas 100 amostragens por segundo.

Se você pretende gravar sinais mais rápidos como ondas acústicas, será necessário então, um algoritmo de conversão mais rápido.

Conversores de aproximação sucessiva

A figura 8.11 mostra o diagrama esquemático de um conversor de alta velocidade de 8 bits. Este circuito é capaz de amostragens da ordem de 200.000 por segundo. Para se obter esta velocidade, a técnica usada é a chamada aproximação sucessiva.

Como no conversor do tipo contador de rampa binário, este A/D usa um conversor D/A em um elo de realimentação, mas substitui os contadores por um circuito especial chamado SAR (registrador de aproximação sucessiva). A lógica do SAR é melhor explicada no diagrama de blocos da figura 8.12.

Inicialmente a saída do SAR e do D/A estão em zero. Depois de um pulso de início de conversão, o SAR permite os bits do D/A. A cada bit, o comparador dá uma saída mostrando se a entrada é maior ou menor do que a saída do D/A. Se a saída do D/A é maior do que o sinal de entrada, um zero é colocado naquele bit em particular. Se for menor fará com que o bit seja 1. O registrador se move sucessivamente para o próximo bit menos significativo (guardando o resultado dos bits já testados).

Depois que todos os bits do D/A forem testados, o ciclo de conversão estará completo. Ao contrário dos 256 clocks do método do contador binário, toda a conversão leva somente 8 clocks. Uma outra conversão começa no próximo clock se estivermos operando no modo de autoconversão. Para guardar os resultados da conversão usou-se um registro de 8 bits (CI3).

Com um clock de 800 KHz o circuito fará 100.000 conversões por segundo. Como os resultados são automaticamente guardados ño registrador, a conversão é transparente ao computador e pode ser lida em qualquer velocidade. Conversores A/D de alta velocidade estão sujeitos a problemas de lay-out e a componentes. Uma freqüência de amostragem prática seria a de 20.000 por segundo.

Uma aplicação para um conversor A/D rápido

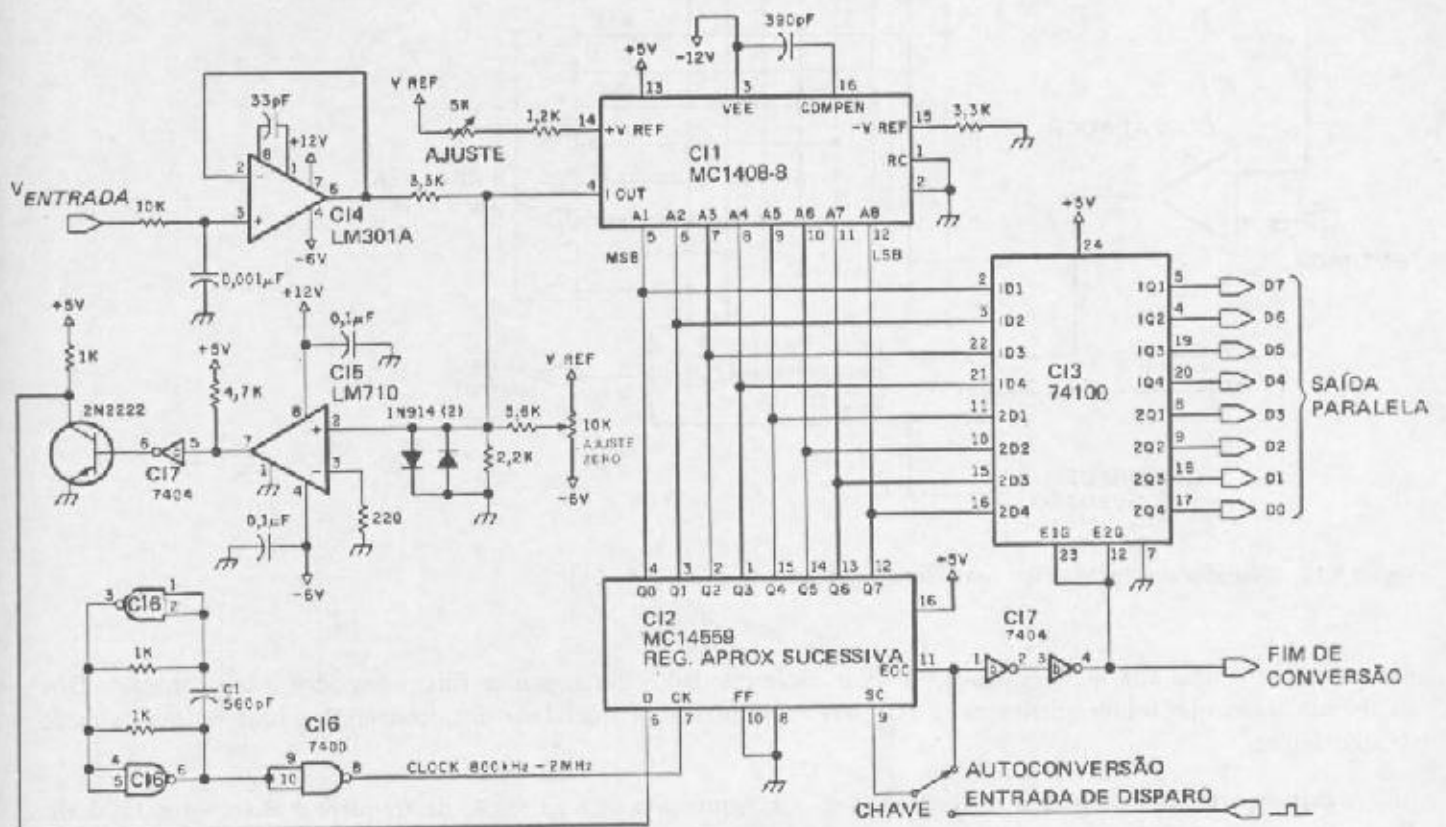
Quando nós consideramos primeiro o uso de conversores com o PAZ, pensamos primeiramente em monitorar algum processo ou transformar o PAZ em um controlador inteligente.

Para isso precisamos de um conversor simples como os já apresentados. Mas com a inclusão de um conversor rápido, algumas experiências à mais poderão ser tentadas, como por exemplo, sinais de audio.

A largura de faixa da voz humana é de 400 Hz. Estes sinais quando falados em um microfone e enviados a um A/D podem ser digitados como qualquer forma de onda. E se as amostragens de voz forem tomadas rapidamente e armazenadas, o dado guardado pode ser usado para reconstruir a mesma voz.

Esta voz reconstituída é chamada fala digitada.

Em essência, a fala digitada é simplesmente o resultado de uma técnica padrão de aquisição de dados.

Quando se fala em um microfone, a sua voz resulta em uma forma de onda, cuja razão de freqüência varia. Se este sinal for aplicado à entrada de um conversor A/D rápido e as conversões armazenadas na memória, o computador

CI#	TIPO	+5V	GND	+12V	-12V	-6V
1	MC1408-8	13	1		3	
2	MC14559	16	8			
3	74100	24	7			
4	LM301A			7		4
5	LM710			8		4
6	7400	14	7			
7	7404	14	7			
8	MC1408-8	13	1		3	
9	LM301A			7	4	

NOTAS:

1. TODOS OS RESISTORES 1/4W 5% CASO NÃO SEJAM INDICADOS.
2. TODOS OS CAPACITORES SÃO 100V CERÂMICA, CASO NÃO SEJAM INDICADOS.
3. COM OS COMPONENTES INDICADOS A FREQÜÊNCIA DO CLOCK Z É 800 KHz. ISTO É 100.000 CONVERSÕES POR SEGUNDO NO MODO AUTOCONVERSÃO.
4. O CIRCUITO A SEGUIR PODE SER ACRESCENTADO A CADA PINO DE SAÍDA DO CI3 SE FOR NECESSÁRIO UMA INDICAÇÃO VISUAL

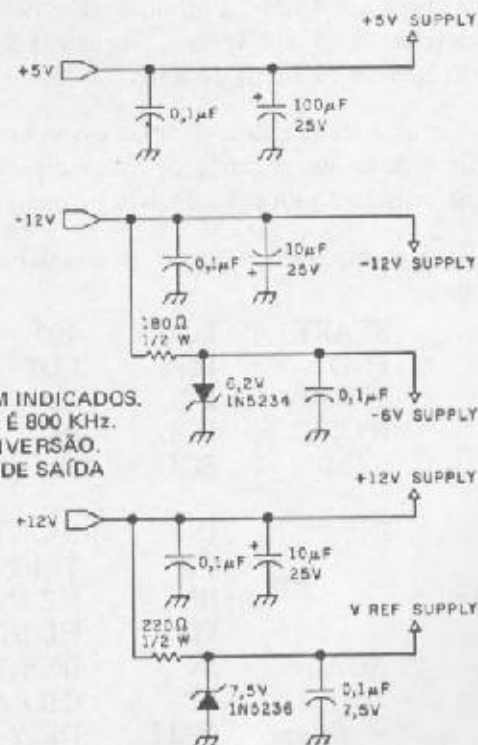
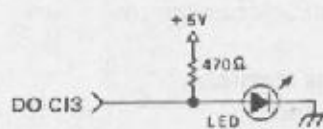



Figura 8.11 Diagrama esquemático de um conversor de 8 bits do tipo aproximação sucessiva.

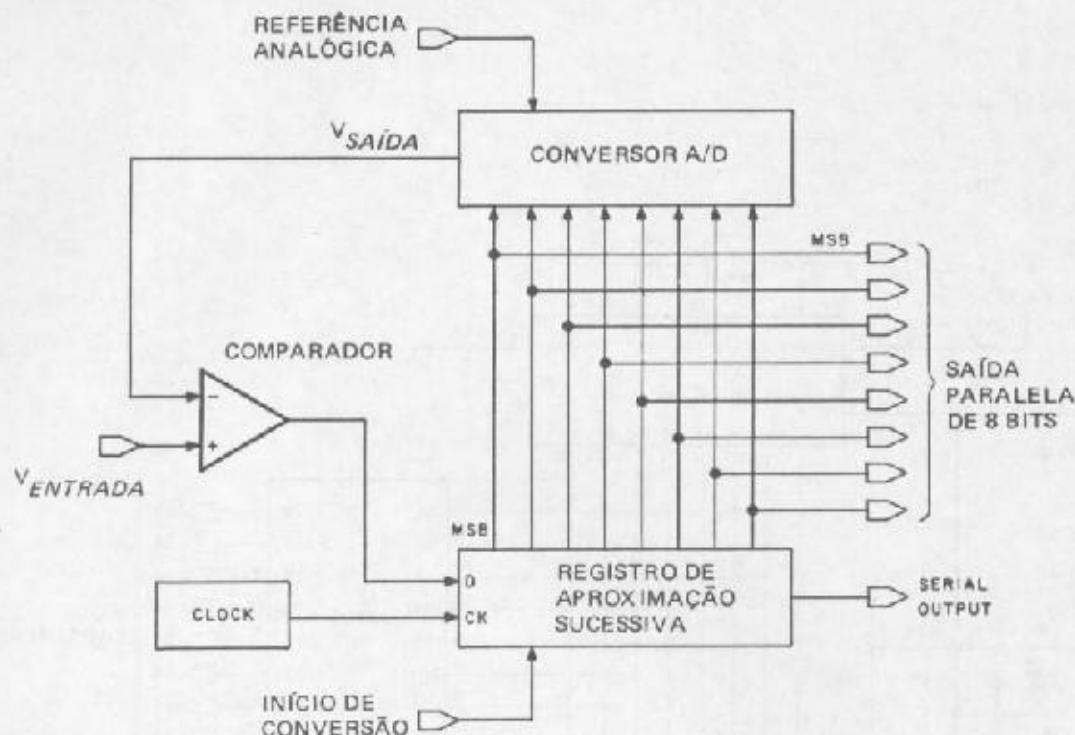



Figura 8.12 Diagrama em blocos de um conversor de aproximação sucessiva de 8 bits.

não saberá se é uma voz ou uma reação nuclear. Se estes dados armazenados forem enviados a um conversor D/A na mesma razão que foram guardados, a fala será reproduzida. A fidelidade de reconversão é uma função da razão de amostragem.

A maior parte da informação do conteúdo da fala humana ocorre na região de freqüência abaixo dos 1500 Hz. Existe uma lei conhecida como "Critério de Nyquist" que é usada para determinar a melhor razão de amostragem. Na teoria esta lei determina que a mínima razão de amostragem deve ser duas vezes a freqüência de entrada. Então, se a voz humana chega aos 4 KHa, a mínima razão de amostragem seria de 8000 por segundo. Em realidade a razão de amostragem deve ser de 3 a 4 vezes a freqüência de entrada. Para se digitar a voz com precisão necessita-se de uma razão de amostragem de 12 KHz a 16 KHz.

Para se usar esta técnica deve-se levar em conta a grande quantidade de memória necessária. Em uma razão de amostragem de 4 KHz, um segundo de fala ocupa 4000 bytes de memória. Se você aumentou mais 2K de memória na configuração original do PAZ, você talvez irá querer experimentar a fala digitada.

Um pequeno exemplo do processo de coordenar a digitação e da armazenagem dos dados. É mostrado a seguir.

```
START   EQU    400         Tabela de início de endereço
END     EQU    C00         Tabela de fim de endereço
TRIG    EQU    A8          Início de conversão de nível
IPORT   EQU    04          Porta de entrada do A/D
SAMP    EQU    38          Tempo de razão de amostragem

INP     IN     IPORT       Lê entrada do A/D
        CP     TRIG        Compara entrada com nível de trigger
        JP     NZ,INP      Refaça-se nível de trigger está abaixo
        LD     HL,START    Carrega tabela de início de endereço
AGAIN   IN     IPORT       Amostre
        LD     (HL),A      Guarde a amostra na memória
        CALL   DELY        Retardo entre amostras
        INC    HL
        LD     A,H
        CP     END         Teste se é fim de tabela
        JP     NZ,AGAIN    Se não, faz outra amostragem
        HALT
```

```
DELY    LD     B,SAMP    Inicie o tempo de retardo
DCR     DEC    B
        JP     NZ,DCR
        RET
```

Quando o programa for executado, ele irá ler a porta de entrada do A/D e irá comparar a leitura com A8H (que é 65% do máximo da escala).

Quando a fala estiver presente, o nível de audio presumivelmente irá exceder este nível. Quando isto acontecer, o programa coloca o endereço da tabela de dados e começa a transferir as amostras na razão de 4000 por segundo. A razão é determinada pelo valor de "SAMP". Quanto maior o número, menor a freqüência de amostragem. Quando a tabela estiver cheia, o programa pára e a memória conterá a representação digitada do que foi falado.

Para se ouvir o dado guardado, use o programa esboçado na seção de conversores D/A. Coloque os limites da área de memória na tabela, escolha então uma constante de tempo que seja a mesma da amostragem. Como a fala digitada é uma aplicação especial do D/A, o circuito deve ser modificado ligeiramente para incluir um filtro passa-baixa. Isto irá melhorar a qualidade do som. O circuito modificado é mostrado na figura 8.13.

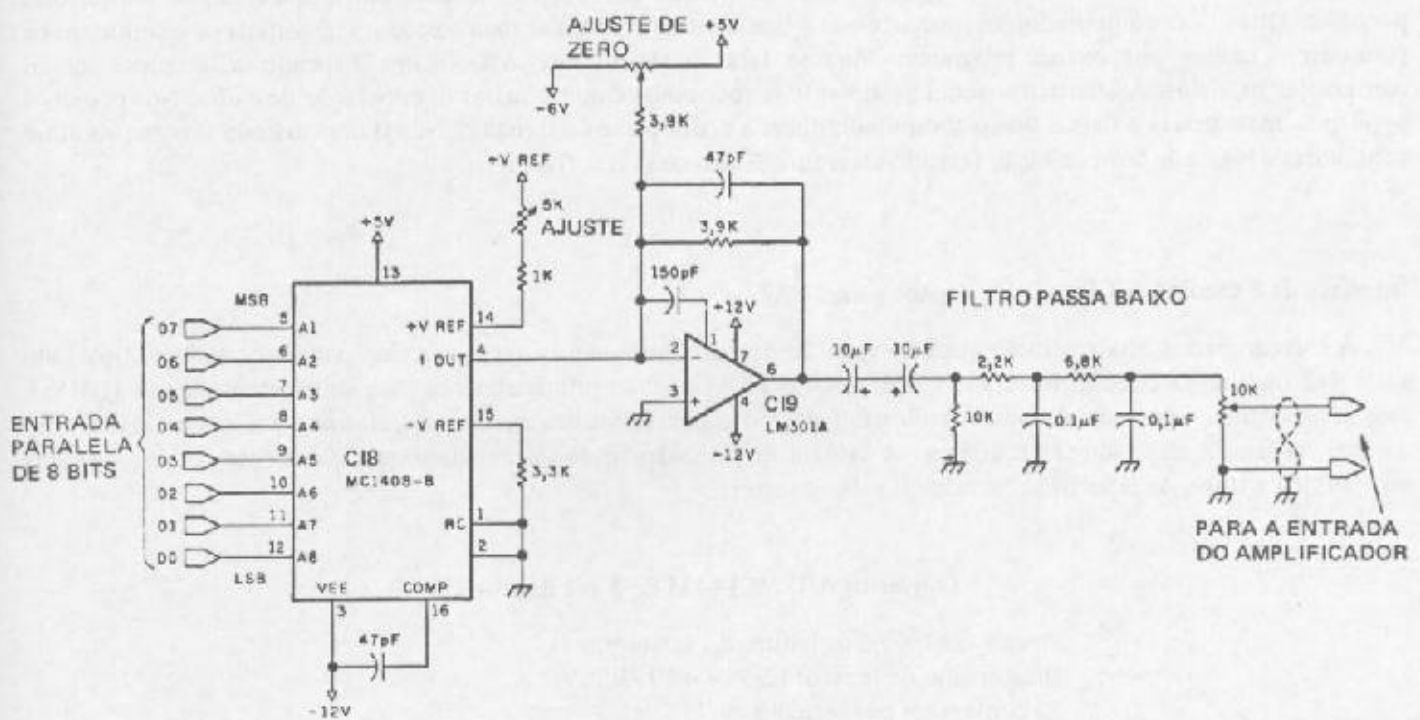



Figura 8.13 Conversor D/A de 8 bits com filtro passa–baixa.

Usando o PAZ como um sistema de aquisição de dados de alta resolução

Os conversores A/D até agora apresentados têm uma resolução limitada e são de um único canal. Eles são adequados para medir a temperatura de um aquecedor solar, mas não têm capacidade de medir um gradiente de temperatura ao longo de um duto de calor. Os sensores usados para medir tais parâmetros precisariam de uma resolução maior do que os sensores de temperatura ambiente. Para uma escala de -20 a 108°C, um conversor de 8 bits teria uma resolução de 0,5°C. Em um aquecedor solar, considerando as variações do movimento do ar, nuvens, esta seria a máxima resolução que você precisaria. Dentro do sistema existem, entretanto, áreas que necessitariam de uma medida mais apurada; um sistema solar é um exemplo típico. Depois da instalação, o próximo passo é investigar como melhorar sua eficiência. Na maioria das vezes consiste em cortar as perdas nos canos e dutos. Uma maneira para se determinar estas perdas é colocar sensores de temperatura ao longo da distribuição de calor, e procurar por regiões frias.

A diferença entre as medidas dos sensores pode ser muito pequena, uns poucos décimos de grau, mas a soma das perdas pode ser significante. Para se medir décimos ou centésimos de um grau, e manter a mesma escala, necessitamos de mais de 8 bits de resolução. Alguma coisa entre 10 e 12 bits é necessário.

A situação torna-se mais complicada pelo grande número de pontos que serão monitorados no sistema. É raro se achar somente um indicador de temperatura em um sistema. No mínimo existiriam seis, ar interno, ar externo, topo do tanque, fundo do tanque, coletor e temperatura do ar de distribuição.

Pouquíssimos sistemas de aquisição de dados utilizam um único canal. Normalmente eles vêm com 8 ou 16 canais multiplexados. A entrada de um conversor A/D é chaveada entre os canais e os resultados são compilados e calculados pelo computador. Esta informação pode ser guardada em fita de gravação, transmitida serialmente para outro sistema, ou usada para rodar um display de tempo real; o que cada um faz com o dado é uma função da aplicação do programa.

Existem vários modos de configurar o PAZ para aquisição de dados de alta resolução. Um deles é simplesmente trocar o A/D de 8 bits por um conversor binário de 12 bits. Quando a conversão terminar, estarão disponíveis 12 bits paralelos de dados. Dependendo do conversor escolhido, podem ser necessários ainda muitos componentes analógicos fora da placa, mas o processo é direto. Infelizmente, estes conversores não são o que você poderia chamar de barato. Apesar deles estarem cada dia mais baratos, no momento ainda são consideravelmente mais caros do que os conversores de 8 bits de velocidade similar.

Muitos conversores binários de 12 bits são caros porque são projetados para dar a aparência de conversores paralelos. Quando o computador necessitar dos 12 bits de dados, este faz uma varredura, manipula-os e armazena-os para serem usados por outros programas. Para se fazer uma interface A/D menos dispendiosa usaremos menos conversores paralelos. A alternativa serial geralmente requer mais tempo e maior manipulação de dados. Nós podemos optar pela mais barata e deixar nosso computador fazer a maior parte do trabalho. Nós já demonstramos como eliminar contadores e lógica de temporização fazendo estas funções através do software.

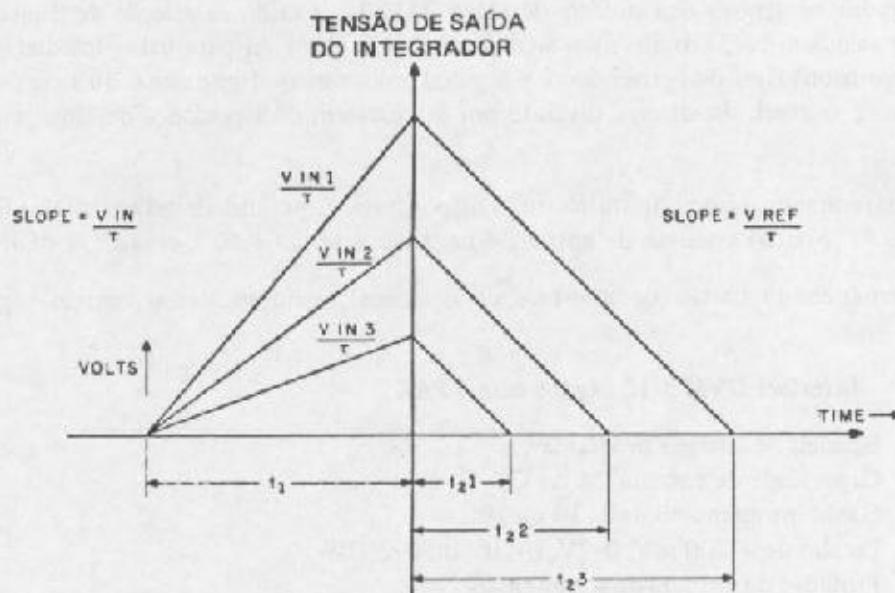
Interface de 8 canais CA/CC de 3 1/2 dígitos para o PAZ

A solução para a alta resolução versus a questão do custo vem em forma de um chip conversor A/D multiplexado de 3 1/2 dígitos. O circuito integrado COMS MC14433 é utilizado principalmente para voltímetros digitais (DMVs), mas satisfaz uma variedade de outras aplicações devido a sua versatilidade. Este é um conversor de um canal com 11 bits, porém, é chamado 3 1/2 dígitos. A saída é BCD (binary-coded decimal) e especificamente cobre uma gama de −1999 a +1999. As especificações básicas estão a seguir:

Conversor A/D MC14433 de 3 1/2 dígitos

Precisão: ±0,05% de leitura ±1 contagem
Duas escalas de tensão: 1,999V e 199,9 mV
25 conversões por segundo
Impedância de entrada 1000 MΩ
Auto zero
Autopolaridade
Disponibilidade de sinais de sobre, sob, e auto escala.

O MC14433 é um conversor A/D dupla rampa modificado e está descrito na figura 8.14. A seqüência de conversão está dividida em dois períodos: desconhecida e referência. Durante a seqüência V_{in} (entrada desconhecida), a tensão desconhecida é aplicada em um integrador com uma constante de tempo de integração definida para um predeterminado limite de tempo. A tensão de saída do integrador torna-se, então, uma função da entrada desconhecida. Quanto mais positiva for a entrada, maior será a saída do integrador.

Durante o segundo ciclo da seqüência, um sinal de referência de 2,000V é conectado em V_{in}. Isto faz com que o integrador mova-se em direção a zero enquanto o circuito digital do chip mantém sua temporização. A diferença de tempo entre as duas seqüências de integração é então uma função de suas diferenças de tensão. Se 2,000V fosse a tensão aplicada V_{in}, então t_2 seria igual a t_1. A tensão desconhecida é equivalente à razão dos períodos de tempo da tensão de referência (V_{REF}). O fundo de escala do conversor é determinado por V_{REF}. Mudando V_{REF} para 0,200V fará com que a contagem de saída de 1999 represente 199,9 mV em vez de 1,999V em fundo de escala.

τ = CONSTANTE DE TEMPO DE INTEGRAÇÃO
t_1 = PERÍODO DE INTEGRAÇÃO (C^{te}) DA TENSÃO DESCONHECIDA
t_2 = PERÍODO DE INTEGRAÇÃO (VARIÁVEL) DA TENSÃO DE REFERÊNCIA

$$V_0 = \frac{V_{IN}}{\tau} t_1 = \frac{V_{REF}}{\tau} t_2$$

QUE É:

$$\frac{V_{IN}}{V_{REF}} = \frac{t_2}{t_1}$$

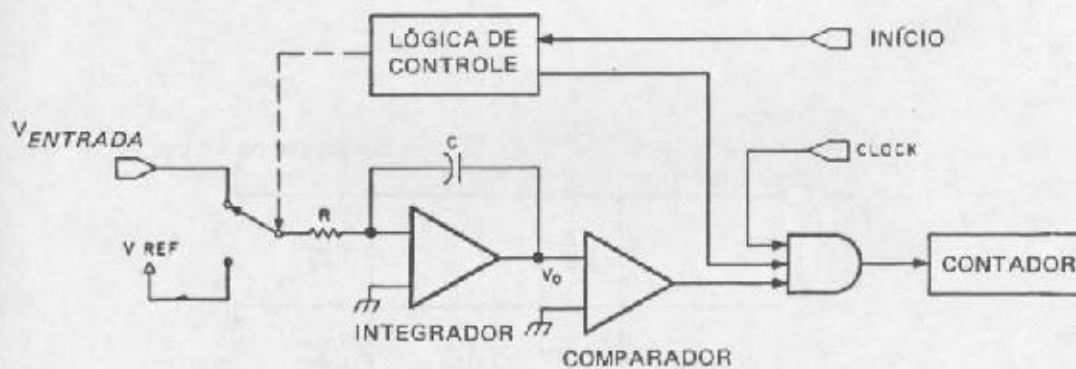



Figura 8.14 Representação simplificada de um conversor A/D dupla rampa.

A saída do chip DVM é uma combinação de dados serial e paralelo. Existem 4 selecionadores de dígito de 4 linhas de dados BCD:

Linhas de saída BCD

Pino 23	Q3 (MSB)
Pino 22	Q2
Pino 21	Q1
Pino 20	Q0

Saídas de seleção de dígito

Pino 19	DS1 (MSB)
Pino 18	DS2
Pino 17	DS1
Pino 16	DS0

Com respeito ao que o computador vê através dos buffers de saída 74LS04, a saída de seleção de dígitos será baixa quando o respectivo dígito for selecionado. O dígito mais significativo (1/2 DS1) vai para baixo imediatamente após um pulso de EOC (end-of-conversion) (fim de conversão) e é seguida pelos outros dígitos na seqüência de MSD para LSD. O clock do multiplexador é o clock do sistema dividido por 80; existem dois períodos de clock entre as saídas de dígitos.

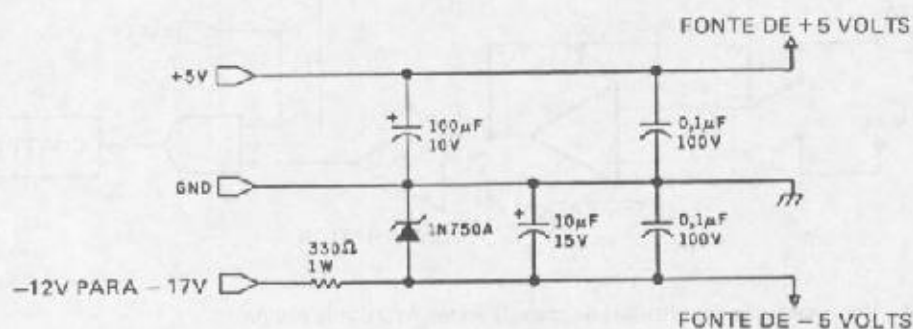
Durante DS1, a polaridade e determinados dígitos de *status* estão disponíveis. A polaridade está em Q2 e o valor do 1/2 dígito está em Q3. Se Q2 é "1", então a tensão de entrada é negativa, e se Q3 é "0", então 1/2 dígito é 0.

A figura 8.15 mostra o esquemático do cartão de interface de 8 canais; como mostrado tem as seguintes capacidades:

Interface DVM 3/12 dígitos para o PAZ

- 8 canais de entrada programáveis
- Capacidade de entrada CA ou CC
- Ganho programável de 1, 10 ou 100
- Escalas de 0-200 mV, 0-2V, 0-20V ou 0-200V
- Proteção de entrada de sobretensão

O CI é o chip DVM MC14433. Este permite aproximadamente 25 conversões por segundo e todas as saídas são reforçadas para fornecer correntes. O CI2 é um chip para precisão da tensão de referência que fornece o sinal V_{REF}. É nominalmente de 2,5V e é ajustado para 2,000V e 0,200V com dois potenciômetros. Apesar do diodo zener poder fornecer a mesma tensão, a variação de temperatura associada com tais componentes torna-os inadmissíveis nesta aplicação.

O CI 5 está configurado como um flip-flop set/reset. Quando a conversão está terminada, um sinal EOC arma CI 5, indicando para o computador que o dado está disponível. Quando o computador termina a leitura do dado, este desarma o flip-flop e aguarda a próxima conversão.

CI	TIPO	+5V	−5V	GND
1	MC14433	24	12	13
2	MC1403	1		3
3,4	74LS04	14		7
5	7474	14		7
6,7	CD4053	16	7	8
8	CD4051	16	7	8
9	7445	16		8
10	LM324	4	11	

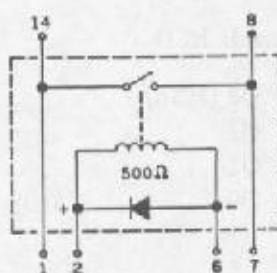



Figura 8.15 Interface de 8 canais de 3 1/2 dígitos 0-200V CA/CC DVM

* 430K E 13K EM SÉRIES
2,500V
10K
2,000V
6,8K
2K
0,200V
CI6
CD4053
CI2
MC1403
V REF
CI6
CD4053
0,1μF
MYLAR
100V
68K
27K
443K*
CI1 MC14433
3 1/2 DÍGITO DVM
0,1μF
100K
0,0022μF
CONVERSOR
AC PARA DC
CI6
CD4053
CI7
CD4053
1M
1N750A
1N750A
X100
11.11K
X10
111K
SE NECESSÁRIO
R TRIM
R TRIM
CI8
CD4051
CI9
7445
CI4
74LS04
CI3
74LS04
CI5
7474
ENTRADAS
#1
#2
#3
#4
#5
#6
#7
#8
+5V
RL1
RL2
RL3
RL4
RL5
RL6
RL7
RL8
ENTRADA PORTA 1
ENTRADA
PORTA 4
SAÍDA PORTA 1
SAÍDA PORTA 1

Os CIs 1,2, 3 e 4 constituem um conversor de um canal de 3 1/2 dígitos. Este tem uma escala de 0,200V ou 2,000V determinada pela V_{REF}. Para executar operação multicanal e capacidade CA, é necessário colocar um multiplexador de entrada e um conversor CA para CC na frente do CI1.

A figura 8.16 mostra a tensão de referência e a seleção de escala desta interface. O MC14433 pode cobrir de 0-199,9 mV ou 0-1,999V. As escalas dependem do nível da V_{REF}. Quando B5 da porta 1 está baixo, as chaves 5 e 6 estão nas posições mostradas. Isto fornecerá 2,000V para V_{REF} e colocará a constante de tempo de integração com um resistor de 82K Ω. Com B5 = 0, V_{REF} é 0,200V e o resistor de integração é 10K Ω.

A figura 8.17 ilustra o subsistema em termos simplificados. SW1 e SW2 representam a seção de seleção de ganho. Como mostrado, o ganho é 1 e nenhum circuito divisor está habilitado. Quando um relé de entrada está fechado (controlado através de CI9), a tensão de entrada daquele canal é enviada diretamente para a entrada de CI1 através de um resistor de 1M Ω. Se a interface está para CC e ganho de 1, um sinal de entrada, 1,400V no canal 3 poderá ser lido diretamente como 1,400V pelo chip DVM. Se, entretanto, fosse aplicado repentinamente 150V, este seria seguro por Z1 e Z2, que protegem CI1. O dado lido pelo computador indicará uma condição de fora de escala porque a entrada estará presa em 4V.

Fechando SW1 ou SW2 forma-se um divisor que permite ao computador ler estas tensões mais altas. Um divisor 10:1 é formado pelo fechamento de SW1. O resultado é um divisor formado pelo resistor R1 de 1M Ω, e um resistor R2 de 111K Ω para terra. O programador deve ter em mente que um divisor foi usado naquele canal e a resposta deve ser multiplicada por 10.

Fechando SW2 forma-se um divisor 100:1. A matemática é a mesma, exceto que o resistor (R3) é agora de 11,11K Ω. Uma entrada de 8V torna-se 0,080V e uma entrada de 150V torna-se 1,500V. Obviamente, a seleção apropriada da escala é necessária para maximizar a resolução.

Uma vantagem adicional desta interface é a capacidade para entradas CA. Isto é possível pela simples conversão do sinal CA para CC após a saída da seção do divisor. CI6 e CI7 funcionam como chaves de um pólo, duas posições para chavear a entrada ou saída de sinal do conversor. O conversor real CA para CC está mostrado na figura 8.18. Este componente é conhecido como conversor RMS (Root Mean Square). Se você aplicar nele um sinal CA de 1,0V de pico, dará uma saída CC de 0,707V. Esta é a técnica usada em muitos multímetros digitais. Este também é o modo como nós comumente expressamos tensões CA. Por exemplo, a tensão de 115CA de nossas casas é uma tensão 115V RMS. O pico é em torno de 176V. O conversor passa CA e CC porque não existe nenhum capacitor na entrada. Se este for inadvertidamente chaveado para um sinal CC, este multiplicará a leitura por 1,414.

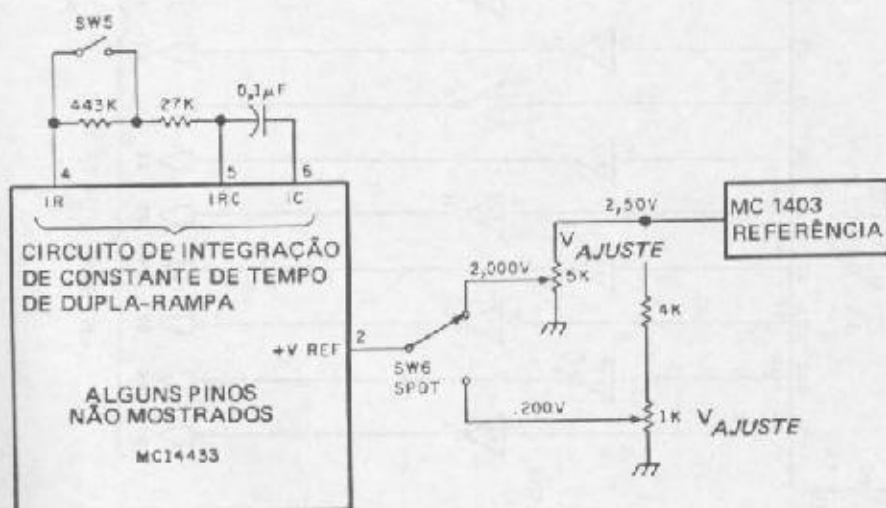



Figura 8.16 Circuito modificado para a tensão de referência e constante de tempo de integração para o voltímetro digital.

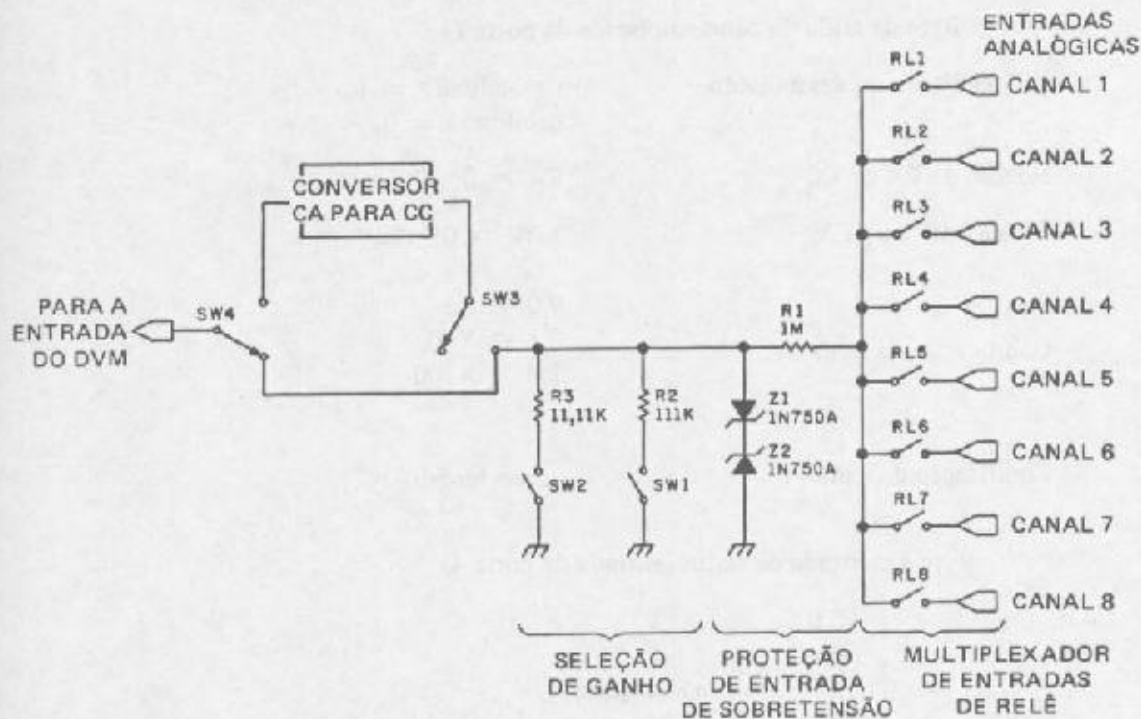



Figura 8.17 Seções condicionadas de entrada do DVM.

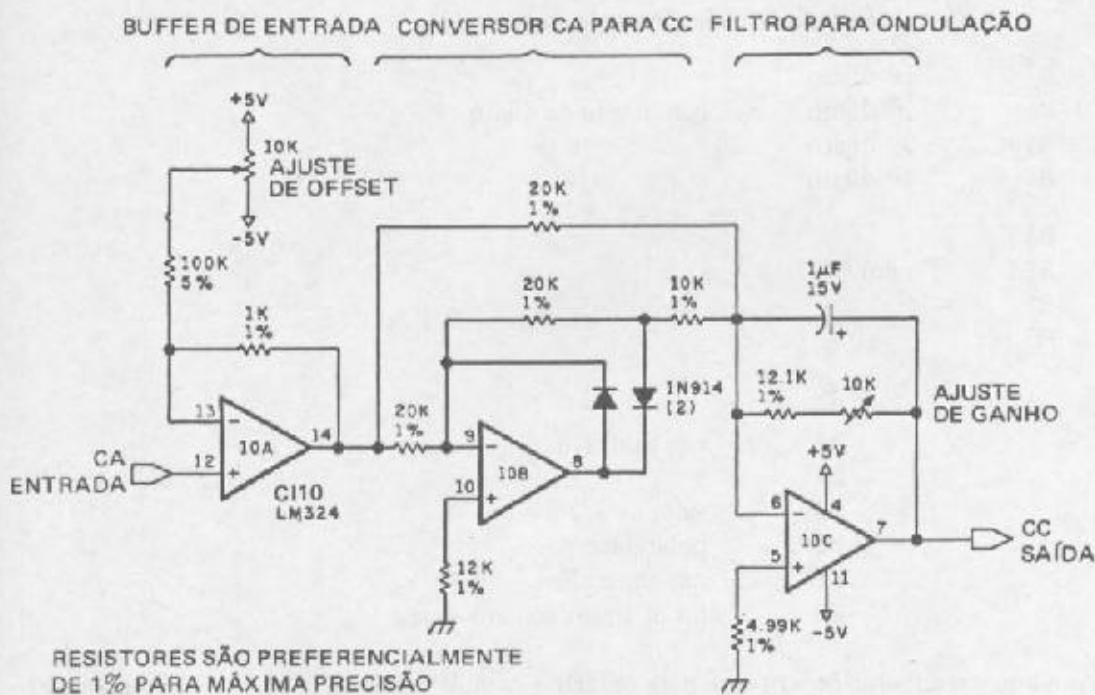



Figura 8.18 Diagrama esquemático de um conversor CA para CC.

Exercitando a interface com um módulo exercitador software

A interface está ligada ao PAZ através das portas de E/S. Utiliza 10 bits de entrada e 8 bits de saída para operação completa. Elas são arbitrariamente escolhidas como portas 1 e 4 para esta descrição. A escolha dependerá de que endereço você ligar quando configurar o PAZ. Estas portas não foram usadas para nada na descrição original e será necessário adicionar o hardware apropriado. Em suma, são as seguintes necessidades de E/S para a interface DVM (voltímetro digital):

Byte de saída de comando (saída da porta 1)

B7	**EC habilitado ou desabilitado**	Desabilitado = 1, habilitado = 0
B6	**Seleção de CA ou CC**	CA = 0; CC = 1
B5	**Escala 2,0V ou 02,V**	2,0V = 0; 0,2V = 1
B4 B3	Codificação do ganho	0,0 = X1 0,1 = X10 1,0 = X100
B2 B1 B0	Codificação do canal	canais binário 0-7

Byte de entrada de status (entrada da porta 4)

B7 B6 B5 B4 B3 B2	não utilizados
B1	fora de escala
B0	fim de conversão

Byte de entrada de dados (entrada da porta 1)

B7	1º dígito	
B6	2º dígito	habilitação de dígito
B5	3º dígito	
B4	4º dígito	
B3 B2 B1 B0	valor BCD	

Quando B7 = 0 então:

B6 B5 B4	não utilizado
B3	valor de 1/2 dígito
B2	polaridade
B1	não utilizado
B0	bit de *status* de auto-escala

Esta interface usa um módulo exercitador de software para reduzir a complexidade de hardware. O programa não é como um módulo exercitador de comunicações. Para obter efetivamente o dado da interface, o computador deve ser sincronizado com o chip DVM e deve executar uma seqüência de operações específicas para demultiplexar o fluxo de entrada de dados.

O programa que faz a interface e armazena os valores do chip DVM é escrito como uma sub-rotina. Todas as informações necessárias para a apropriada execução do módulo exercitador estão no par de registros DE na hora da chamada. Seu conteúdo dirá à interface qual canal ligar, se este será CA ou CC, e qual V_{REF} e ganho utilizar. Um canal é verificado toda vez que a sub-rotina for chamada.

A informação colocada no par de registros DE na hora da chamada é o byte de saída de comando (saída da porta 1) e cada bit tem o destino listado anteriormente. A única diferença é que o bit 7 (o bit de habilita/desabilita para o conversor A/D) é enviado como um lógico 0 quando faz a chamada. O módulo o colocará na condição de habilitado após tê-lo posto no devido relé e aguardado um retardo de 1,3 ms.

A demultiplexação da saída do chip DVM é direta. Após a chamada, as saídas para a interface fecham as chaves apropriadas, e o processador central entra em um ciclo de espera do sinal de fim de conversão. Quando isto acontece o programa sabe que os próximos 4 dígitos de dados é o que quer o processador. O chip DVM liga cada uma das linhas de seleção de dígito sucessivamente, e o programa grava o valor das 4 linhas de dado BCD de cada vez. Este obtém os bits de *status* e polaridade a partir do MSD do byte 1/2 *dígito* e reformata, e armazena o valor da tensão de entrada em 4 bytes da memória. Os 3 dígitos inteiros são armazenados na notação BCD e ocupam 3 bytes. O 1/2 dígito, polaridade e indicação de fora de escala estão localizados no quarto byte. A polaridade é indicada através do MSB. Uma leitura positiva corresponde ao lógico 1 e uma negativa ao lógico 0. O valor do 1/2 dígito só pode ser 0 ou 1 e ocupa o LSB da quantidade. A indicação de fora de escala é manuseada com uma pequena manipulação de programa. Se o módulo deteta que a leitura da entrada não está dentro da escala, este coloca o equivalente de +2 no byte de 1/2 dígito. Obviamente, esta é uma condição ilegal para um DVM capaz apenas de contar até 1999. O programador usando este dado armazenado poderá verificar os limites do dado antes de agir sobre ele.

Quando o módulo completa sua operação, terá ativado uma leitura de 3 1/2 dígitos armazenando-os como 4 bytes em uma tabela especial na memória. Os 8 canais de dados constituem uma tabela de 32 bytes. A posição de um determinado canal de dados é encontrada através de uma simples expressão:

Os 4 bytes de dados iniciam na posição de memória

$$L + 4(N-1)$$

onde L = endereço inicial da tabela de memória
N = número do canal (1 a 8)

A figura 8.19 é a listagem do programa que exercita esta interface DVM. Quando notado, este ocupa menos do que uma página da memória.

Nota: Deve-se ter em mente a precaução quando se medir sinais CA com esta interface. O terra da interface DVM é o mesmo do computador; um curtocircuito de potencial existirá, a menos que a fonte do computador ou a tensão medida sejam isolados.

```
L----!--------!-----------------!------------------------------
0100 *
0110 *** MC14433 DRIVER CONVERSOR A/D DE 3 1/2 DIGITOS
0120 *
0125 * REV. 1.9
0130 *
0140 DIP      EQU  1              NUM.DA PORTA DE ENTRADA DE DADOS
0150 SIP      EQU  4              NUM.DA PORTA DE ENTRADA DE STATUS
0160 COP      EQU  1              NUM.DA PORTA DE SAIDA DE COMANDO
0170 EEOC     EQU  200            HABILITA ENTRADA EOC
0180 DEOC     EQU  000            DESABILITA ENTRADA EOC
0190 *
0200 *
0210 * BUFFERS DE DADOS DOS CANAIS
0220 *
0230 CHAN0    DW   000000
0240          DW   000000
0250 CHAN1    DW   000000
0260          DW   000000
0270 CHAN2    DW   000000
0280          DW   000000
0290 CHAN3    DW   000000
0300          DW   000000
0310 CHAN4    DW   000000
0320          DW   000000
0330 CHAN5    DW   000000
0340          DW   000000
0350 CHAN6    DW   000000
0360          DW   000000
```

Figura 8.19 Listagem de um programa em linguagem assembly que exercita o voltímetro digital.

```
0370 CHAN7    DW   000000
0380          DW   000000
0390 *
0400 * BUFFER DE DADOS INTERMEDIARIOS
0410 *
0430 CHAN     DB   000          NUM. DO CANAL EM CURSO
0440 CCP      DW   000000       COMANDA PARAMETRO DO CANAL
0460 *
0470 *
0480 *** INICIA CONVERSOR A/D
0490 *
0550 *
0560 START    LD   A,E
0570          LD   (CCP),A
0580          AND  007
0590          LD   (CHAN),A
0600          LD   IX,CHAN0
0910          LD   D,0
0920          LD   E,A
0930          SLA  E            CALCULA DIFERENCA DE BUFFER
0940          SLA  E
0950          ADD  IX,DE
0960 *
0970 * SELECIONA CANAL E INICIA CONVERSAO
0980 *
0985          LD   B,3          SETA CICLO DE CONTAGEM
0990 SCSC     LD   A,(CCP)
1000          OUT  COP          SELECIONA CANAL
1005          CALL DELAY
1010          OR   EEOC         HABILITA SAIDA EOC
1020          OUT  COP          COMANDA CONVERSOR A/D
1030 *
1040 * ESPERA POR EOC
1050 *
1060 WEOC     IN   SIP          LE STATUS DO CONVERSOR
1070          BIT  0,A          TESTA PARA EOC
1080          JR   Z,WEOC       SALTA SE NAO OK
1085          DJNZ SCSC
1090          BIT  1,A          TESTA PARA SOBRE ESCALA
1100          JR   NZ,OVER      SALTA SE VERDADEIRO
1110 *
1120 * TERMINO DA CONVERSAO; PROCESSA PRIMEIRO DIGITO (MSD)
1130 *
1140 MSD0     LD   B,200        SELECIONA DIGITO 1
1150          CALL RDIG         ESPERA E LE DIGITO 1
1160          CPL
1170          RRCA RIGHT        POSICIONA VALOR DO DIGITO
1180          RRCA
1190          RRCA
1200          AND  1            ISOLA
1210          LD   E,0          INICIALIZA BYTE DE STATUS
1220          BIT  2,D          TESTA POLARIDADE
1230          JR   NZ,MSD3      SALTA SE POSITIVA
1240          LD   E,200        CARREGA SINAL DE POLARIDADE
1440 *
1450 * SALVA MSD E POLARIDADE
1460 *
1470 MSD3     OR   E            SOMA SINAL POLARIDADE A MSD
1480          LD   (IX+0),A     SALVA NO BUFFER DE DADOS
1500 *
1510 * PROCESSA SEGUNDO DIGITO
1520 *
```

Figura 8.19 Continua

```
1530          RRC  B           SELECIONA DIGITO 2
1540          CALL RDIG        ESPERA E LE DIGITO
1550          AND  017         ISOLA
1560          LD   (IX+1),A    ARMAZENA SEGUNDO DIGITO
1570 *
1580 * PROCESSA TERCEIRO DIGITO
1590 *
1600          RRC  B           SELECIONA TERCEIRO DIGITO
1610          CALL RDIG        ESPERA E LE DIGITO
1620          AND  017         ISOLA
1630          LD   (IX+2),A    ARMAZENA
1640 *
1650 * PROCESSA QUARTO DIGITO
1660 *
1670          RRC  B           SELECIONA QUARTO DIGITO
1680          CALL RDIG        ESPERA E LE DIGITO
1690          AND  017         ISOLA
1700          LD   (IX+3),A    ARMAZENA
1710 RAPUP    RET
1720 *
1730 * CARREGA VALOR DE SOBRE ESCALA 2000 NO BUFFER DE DADOS
1740 *
1750 OVER     LD   A,2         CARREGA VALOR DE MSB
1760          LD   (IX+0),A
1770          XOR  A
1780          LD   (IX+1),A    CARREGA VALORES DE LSD
1790          LD   (IX+2),A
1800          LD   (IX+3),A
1810          JR   RAPUP
1870 *
1880 *
1890 * ROTINA DE LEITURA DE DIGITO
1900 *
1910 RDIG     IN   DIP         LE BYTE DE DADO
1920          CPL              CONVERTE PARA LOGICA ALTA
1930          LD   D,A         SALVA COPIA
1940          AND  B           TESTA SE DIGITO OK
1950          JR   Z,RDIG      SALTA SE NAO
1960          LD   A,D         RESTAURA REGISTRO A
1970          RET              RETORNA A QUEM CHAMOU
1980 DELAY    LD   C,377
1990 DEL1     DEC  C
2000          RET  Z
2010          JR   DEL1
```

Aplicações

Sinto que a aquisição de dados é uma aplicação natural para o PAZ. A interface descrita acima pode ser usada em um sistema de aquecimento solar para monitorar e gravar os dados pertinentes. Usando as facilidades do monitor PAZ e a rotina da interface DVM, pode-se utilizar uma DATA LOGGER (registrador de dados) de 8 canais. Em geral, tudo que seria necessário é um programa supervisor que chame o DVM 8 vezes para obter as entradas de 8 sensores. Este então coloca os limites da tabela de memória para uma sub-rotina de saída serial e armazena as leituras em um cassete. Isto pode ser feito continuamente ou em intervalos de tempo. O sistema poderá incluir um relógio de tempo real, dessa forma as leituras e o tempo em que elas ocorreram poderão ser gravados.

Relógio de tempo real

Se o PAZ for utilizado para aquisição de dados críticos ou controle de funções, deve-se considerar a sincronização de tempo real com o evento do processo. Uma simples definição de sistema de tempo real é a que responde à necessidade de ação em um período de tempo proporcional à urgência da necessidade. Isto recai no fato de que o computador deve ser capaz de executar uma ação específica em um tempo específico. Para isto ocorrer, o computador deve ser capaz de contar o tempo.

Nós podemos realizar isto através de aplicações em hardware ou software. A técnica mais simples é utilizar um circuito de relógio (figura 8.20) para prover o tempo para a linha de interrupção não mascarável do processador central. Isto pode ser a cada 60, 10, ou 1 segundo, como sugerido no esquemático. Quando o computador reconhece a interrupção, primeiro salva todos os registros do programa que estava executando, e então trata da interrupção de tempo real. Freqüentemente, a primeira ação é incrementar um contador interno que mantém controle da hora de interrupção. Normalmente este é um valor equivalente ao número total das batidas do relógio, se em segundos ou milissegundos. Uma vez que este intervalo regular tenha sido estabelecido, é fácil para o computador executar as funções de tempo real.

A resolução do relógio em milissegundos parece maior e torna o intervalo de tempo extremamente mais apurado. Entretanto, eu duvido que muitos construtores do PAZ queiram utilizar tal interface devido à complexidade do software envolvido. Prefiro uma interface que seja fácil de montar e de ser usada.

Essencialmente, o tipo de sistema de tempo real mais adequado aos possuidores do PAZ tem uma resolução de provavelmente 1 minuto em vez de 1 ms. Também, esta é melhor se tiver de ser lido diretamente em horas e minutos em vez de um total da contagem do relógio. O computador não tem de reconhecer a atualização do relógio ou varrer os flags de *status*. À primeira vista, isto pode parecer não ser muita coisa, mas algumas sub-rotinas podem usar até 10% do tempo do processador tratando uma interrupção de milissegundo.

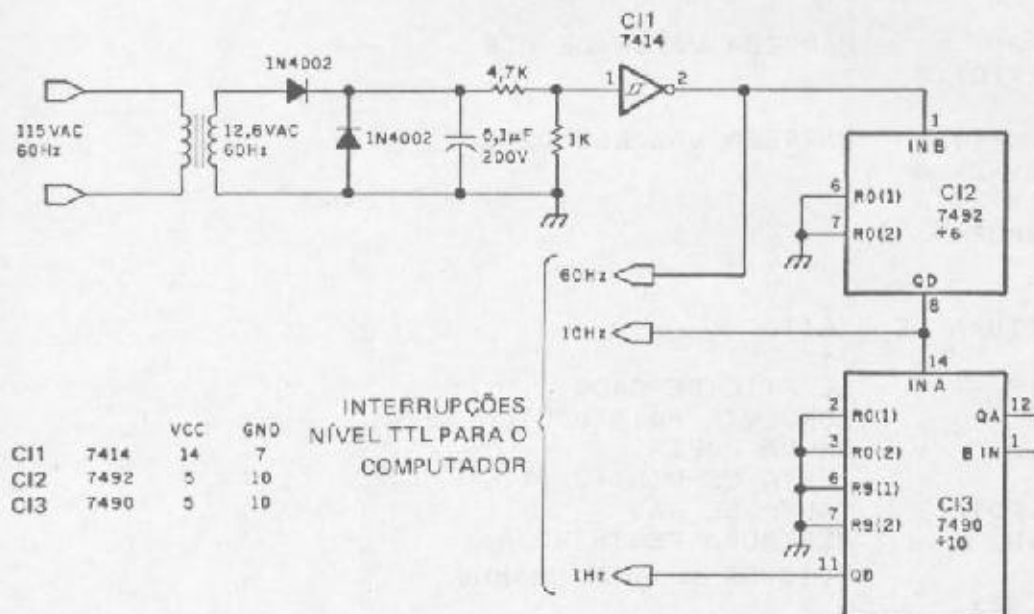



Figura 8.20 Gerador de base de tempo para um módulo de interrupção de relógio de tempo real.

Um velho chip relógio para nos socorrer

O modo mais fácil de se obter uma entrada de hora em hora e minuto a minuto é interfacear o computador com um chip relógio MOS/LSI similar àqueles encontrados em muitos relógios digitais. Existem duas aproximações a um projeto de interface de relógio: um método é deixar o circuito de relógio operar independentemente do computador, ligado de forma que o computador possa monitorar as linhas de saída e extrair o valor da hora a qualquer tempo. O software necessário para este método seria muito parecido com o da interface DVM descrito anteriormente. O outro método, que prefiro, porque envolve menos software, é dar ao computador completo controle sobre o fluxo de informação do relógio de maneira síncrona.

A figura 8.21 mostra esta interface de relógio. Este circuito, manualmente acertado para manter sua simplicidade, é dirigido ao computador. O circuito básico de 4 chips consiste de um chip relógio de saída digital MM5312 4-dígitos BCD/7 segmentos, um gerador de base de tempo MM5369, e dois buffers MOS para TTL para enviar dados para o processador.

A hora é colocada no chip através do aterramento das linhas de slow (lento) e fast (rápido), pinos 14 e 15. Para saber o que está sendo colocado, você deve ler a interface ao mesmo tempo, e mostrar a hora no display de endereço hexadecimal de 4 dígitos, já incluído como parte da expansão do PAZ. A hora é lida através da interface como

números BCD. As 8 linhas de entrada para o computador são ligadas em uma porta de entrada paralela de 8 bits, são divididas em 4 linhas de habilitação de dígito e 4 linhas BCD de valor de dígito. O dado aparece como um dígito habilitado e um número associado BCD. Os décimos de minuto são lidos de B0 a B3 quando B5 está alto (B4, B6 e B7 estão baixo). Da mesma forma, B0 a B3 conterá a quantidade dos décimos de hora quando B7 estiver alto. A lógica da interface permanecerá em um determinado dígito até que seja instruída para proceder o próximo dígito. A seqüência está sobre controle de programa e utiliza um bit de saída de uma determinada porta paralela.

IC#	TYPE	+5V	+12V	GND
1	MM5312		13	23
2	MM5369		8	2
3	CD4049	1		8
4	CD4049	1		8
5	7406	14		7
6	74147	16		8
7	CD4050	1		8

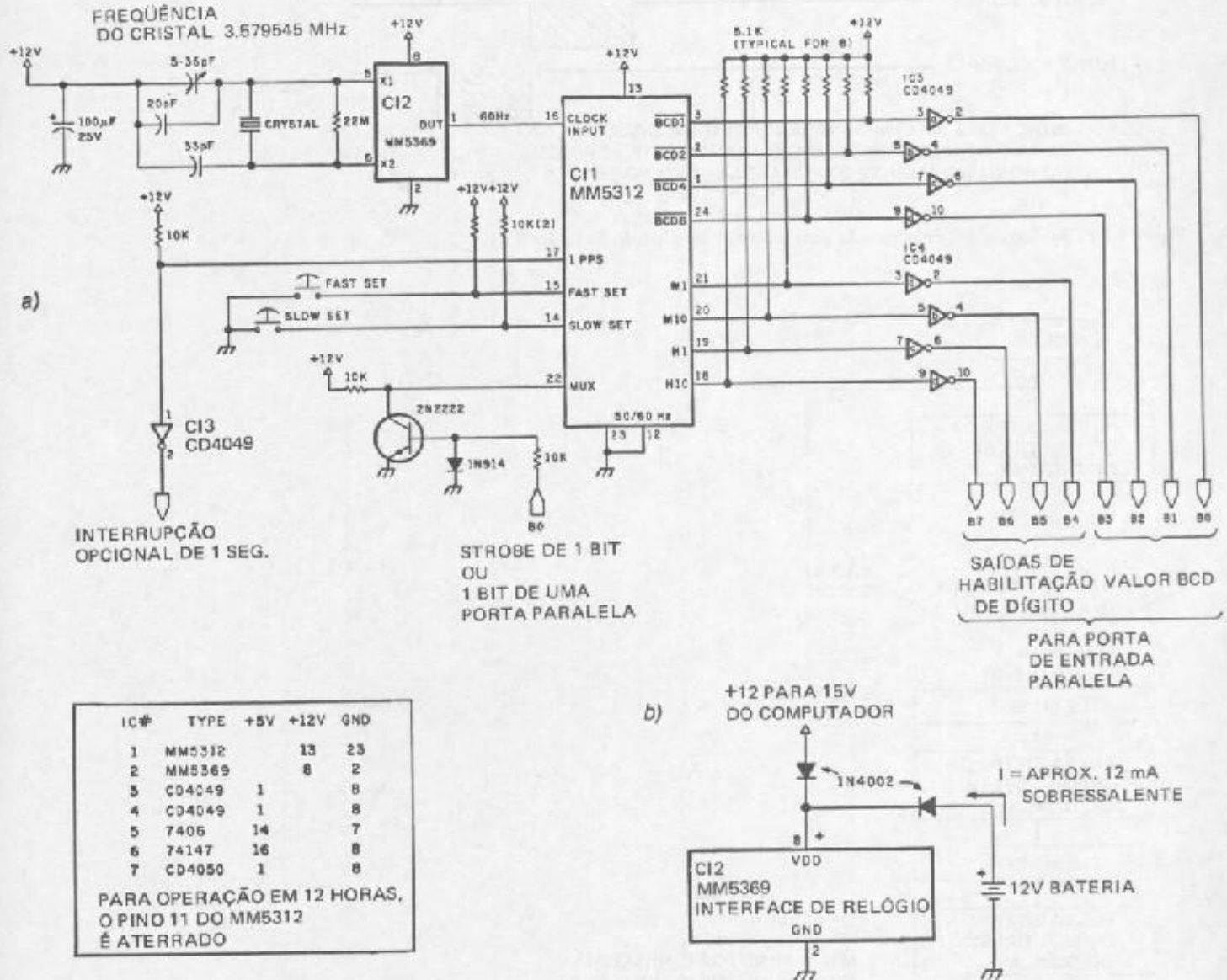



Figura 8.21 Diagrama esquemático de uma interface de relógio de tempo real.
a) Usando um chip de relógio digital MOS.
b) Com bateria sobressalente.

A figura 8.22 mostra como a linha do multiplexador é controlada nesta aplicação. Um bit de uma porta de saída é usado para pulsar o pino de entrada 22 do multiplexador. (Tudo o que se precisa é um pulso de 1 ms. Como uma alternativa, um one-shot pode ser gatilhado por uma linha de strobe decodificada de uma porta não ligada.) A qualquer tempo, 1 das 4 linhas de habilitação de dígito estará baixa e um valor do dígito estará nas linhas de saída BCD. Simplesmente determine que dígito é este e armazene o valor. Em seguida nós pulsamos a entrada do multiplexador para habilitar o próximo dígito e salvá-lo também. É concebível que isto leve somente 4 interações deste procedimento para se obter a leitura completa dos 4 dígitos. Se você prefere uma aproximação mais ordenada, você pode seguir o fluxo do programa descrito na figura 8.23. A única diferença é que este espera até que o chip circule para o início antes de armazenar as leituras.

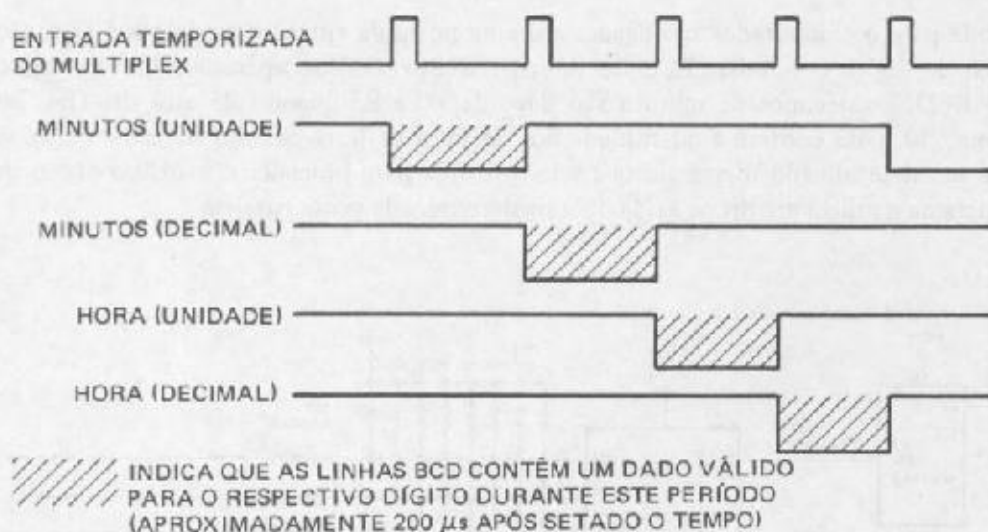



Figura 8.22 Seqüência de temporização para o display do circuito da figura 8.21.

Figura 8.23 Fluxograma do programa para o circuito da figura 8.21

CAPÍTULO 9

CONSTRUA UM TERMINAL TRC

Terminal TRC versátil de baixo custo

Este capítulo descreve o projeto de um terminal TRC (Tubo de Raio Catódico) de baixo custo. Dois componentes MOS/LSI da Standard Microsystems Corporation reduzem o número de partes necessárias para um terminal TRC aumentando ainda sua capacidade.

Os dois componentes, o CRT 5027 temporizador e controlador de vídeo e o CRT 8002 controlador de símbolos do display de vídeo, fornecem virtualmente todo o circuito para a parte de display do terminal TRC (veja apêndices C8 e C9 para especificações).

O terminal é projetado para funcionar sozinho e comunicar-se com qualquer sistema de computador via uma interface RS-232C. Se, no PAZ expandido, o display hexadecimal de 6 caracteres for inadequado, então você tem somente que construir esta unidade e ligá-la na porta serial já montada.

Descrição dos componentes

O CRT 5027 contém a lógica necessária para gerar todos os sinais de temporização (sincronização vertical e horizontal, restauração de endereço de memória paginada etc.) requisitados por um terminal TRC. O formato completo do display incluído entrelace/não entrelace, caracteres por linha, linhas por quadro, varredura por linha, largura de pulso de sincronização horizontal e temporização programável pelo usuário para todos os formatos standard e muitos não standard.

Apesar do CRT 5027 ser estruturado basicamente para uso com seu próprio microprocessador, este projeto descreve um "terminal burro" usando uma PROM de baixo custo e lógica TTL standard para substituir o controle do microprocessador. Mesmo aumentando o número de partes, este projeto resulta em um terminal alfanumérico/gráfico de alta qualidade e baixo custo.

O CRT 8002 fornece uma matriz de pontos de 7 X 11, ROM geradora de 128 caracteres, e um registro de deslocamento de cursor de vídeo de alta velocidade. Este inclui também funções como sublinhar, piscar, mudança de vídeo, espaço em branco e sobre escrita. Adicionais modos gráficos: largo e fino permite a criação de desenhos de linhas, formas e símbolos gráficos únicos.

Descrição do terminal

Como em muitos projetos eletrônicos, um terminal TRC envolve um largo número de performance e custo. Um formato de tela de 16 linhas de 64 caracteres por linha foi selecionado para minimizar as necessidades de memórias (1K bytes) e manter a freqüência de vídeo dentro dos limites de baixo custo dos monitores de vídeo. Uma linha de 80 caracteres não somente aumentaria a freqüência de vídeo abaixo da banda de muitos monitores de baixo custo, como também aumentaria a necessidade de memória. Da mesma forma, maior número de linhas por página aumentará a necessidade de memória, a menos que seja reduzido o número de caracteres por linha.

Em muitas aplicações de microprocessadores, a memória de página é partilhada com o processador através da via de dados. Nesta aplicação, a memória de página é usada estritamente pelo TRC com entrada de dados, caracter por caracter, sincronamente na posição do cursor.

Gráficos completos ou atributos podem ser selecionados em uma base caracter por caracter usando palavras de controle na via de entrada de dados. A figura 9.1 mostra um diagrama de bloco do terminal.

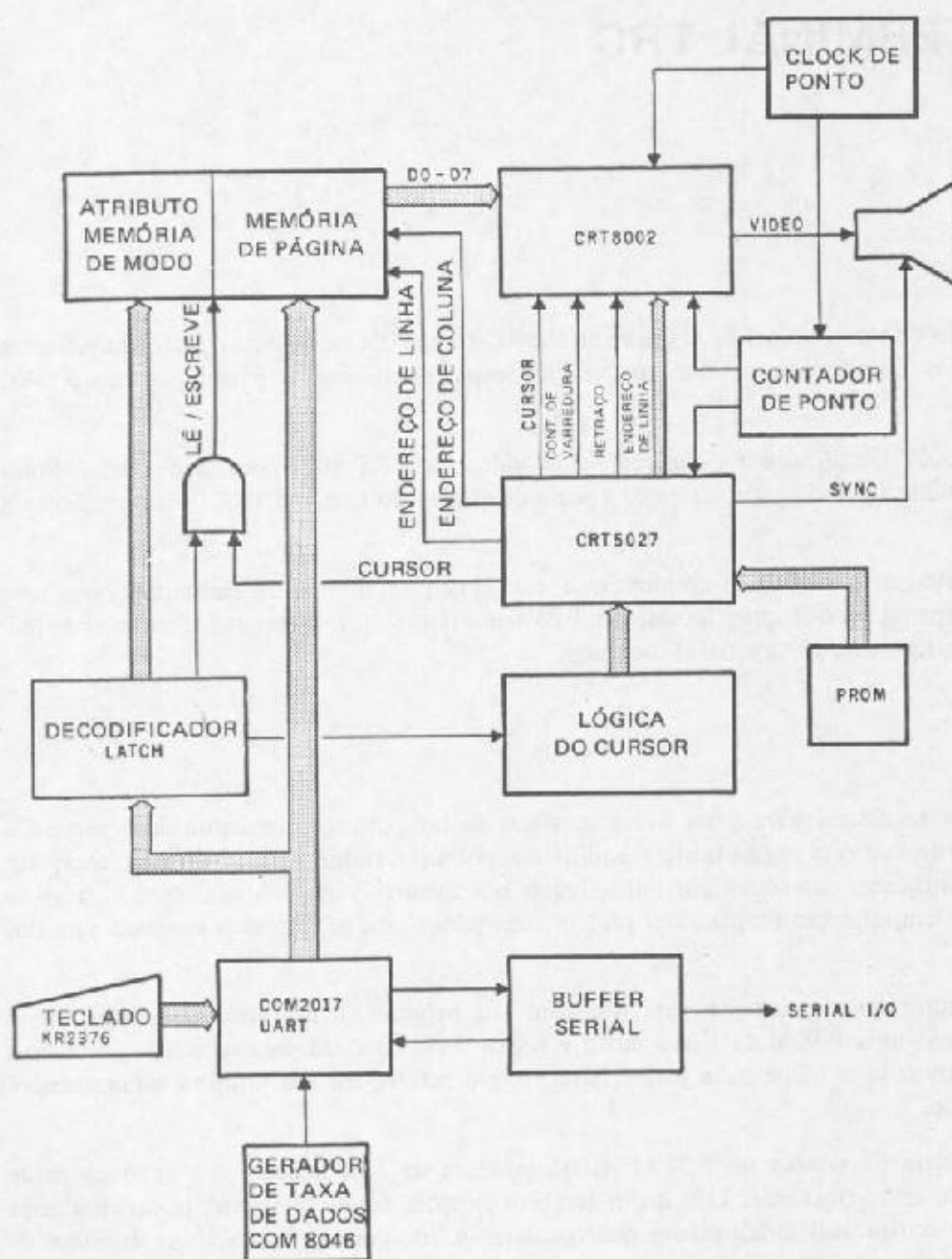



Figura 9.1 Diagrama de bloco de um terminal TRC de baixo custo.

Formato do caracter

O CRT 8002 necessita no mínimo de um bloco para caracter de 8 X 12 para formar seu caracter básico de 7 X 11 e para fornecer linha e espaço de caracter. Entretanto, a fim de permitir enquadrar um caracter completo em uma representação de vídeo reverso, o bloco de caracter horizontal deve ser aumentado para 9 ou 10 pontos. Pela mesma razão, alocando 13 linhas por caracter permitirá muito bem o enquadramento em cima ou em baixo.

Com as varreduras básicas de TV de 60 Hz (vertical) e 15,750 Hz (horizontal), existem 15.750 : 60 = 262,5 linhas por quadro. Como operações de não entrelace necessitam de um número par de linhas, uma freqüência horizontal de 15.720 Hz é utilizada. As 16 linhas multiplicadas por 13 linhas de varredura por linha resultam em 208 linhas de dados mostrados. As 54 linhas remanescentes serão automaticamente colocadas em branco pelo CRT 5027 e formarão as margens superior e inferior.

Para permitir margens esquerda e direita tanto quanto tempo de retraço, um total de vezes para 80 caracteres é alocado por linha. Um bom procedimento é fazer com que o número multiplicativo do caracter seja 25% maior do que o número real de caracteres mostrados.

A freqüência do clock de vídeo é calculado da seguinte forma:

$$10 \text{ (pontos por caracter)} \times 80 \times 15{,}720 \text{ Hz} = 12{,}576 \text{ MHz.}$$

Veja a folha de operação na tabela 9.1.

1. MATRIZ DE CARACTERES HORIZONTAIS (Número de Pontos)	7
2. MATRIZ DE CARACTERES VERTICAIS (Número de Linhas da Varredura Horizontal)	11
3. BLOCO DO CARACTER HORIZONTAL (Passo 1 + Espaço Horizontal Desejado = Nº de Pontos)	10
4. BLOCO DO CARACTER VERTICAL (Passo 2 + Espaço Vertical Desejado = Nº de Linhas da Varredura Horizontal)	13
5. FREQÜÊNCIA DE ATUALIZAÇÃO DE QUADRO (Hz)	60
6. NÚMERO DE LINHAS DE CARACTERES	16
7. NÚMERO TOTAL DE LINHAS DE VARREDURA (Passo 4 + Passo 6 = Nº de Linhas da Varredura Horizontal)	208
8. RETARDO NO SINCRONISMO VERTICAL (Nº em Linhas Horizontais)	26
9. SINCRONISMO VERTICAL (Nº em Linhas da Varredura Horizontal; T = 190,8 μs*)	3
10. RETARDO NA VARREDURA VERTICAL (Nº em Linhas da Varredura Horizontal; T = 1,59 ms*)	25
11. QUADRO TOTAL VERTICAL (Some os Passos de 7 a 10 = Nº em Linhas da Varredura Horizontal)	262
12. FREQÜÊNCIA DA LINHA DE VARREDURA HORIZONTAL (Passo 5 + Passo 11 = Freqüência em Hz)	15.720
13. Nº DE CARACTERES POR LINHA HORIZONTAL	80
14. RETARDO NO SINCRONISMO HORIZONTAL (Nº em Unidade de Tempo de Caracter T = 4,77 μs **)	6
15. SINCRONISMO HORIZONTAL (Nº em Unidade de Tempo de Caracter; T = 5,57 μs**)	7
16. RETARDO NA VARREDURA HORIZONTAL (Nº em Unidade de Tempo de Caracter; T = 2,38 μs **)	3
17. UNIDADES DE TEMPO DE CARACTER EM UMA LINHA DE VARREDURA HORIZONTAL TOTAL (Some os Passos 13 a 16)	80
18. FREQÜÊNCIA DO CARACTER (Passo 12 X Passo 17 = Freqüência em MHz)	1,2576
19. FREQÜÊNCIA DOS PONTOS (Passo 3 X Passo 18 = Freqüência em MHz)	12,576

* Intervalo vertical
** Intervalo horizontal

Tabela 9.1 Folha de operação do CRT 5027 para um formato de tela não entrelaçada de 64 caracteres por linha, 16 linhas.

Programando o VTAC (Video Timer and Controller)

O CRT 5027 VTAC (temporizador e controlador de vídeo) é programado pelo usuário para todas as temporizações e formatos necessários. O dado programado é armazenado em um chip de 9 registros. Embora um microprocessador possa facilmente fornecer o dado programado, uma PROM de baixo custo é utilizada para esta aplicação. Os 9 registros são programados como a seguir (veja tabela 9.2):

REGISTRO 0: Este registro contém o número de "vezes caracteres" para um período horizontal, e é normalmente 1,25 vezes o número de caracteres por linha, neste caso 64 X 1,25 = 80. Como o contador interno é inicializado em zero, o número atual no registro é 80 − 1 = 79.

Registro 0

REGISTRO 1: Este possui 3 campos:

1) Bit 7 – um para entrelace, zero para não entrelace. Neste exemplo selecionamos operação de não entrelace.
2) Bit de 3 a 6 programam o número de "vezes caracter" para a largura do pulso de sincronização horizontal. Este parâmetro é dependente do monitor e é tipicamente de 5 μs. Por existir 80 "vezes caracter" para uma varredura horizontal de 63,6 μs (1 ÷ 15,720), cada vez de caracter é 0,801 μs; 7 "vezes caracter" será usada para gerar um pulso de 5,56 μs.

3) Bits de 0 a 2 posicionam a entrada horizontal. Estes posicionam o dado horizontalmente. As especificações do monitor determinarão a programação inicial, embora algumas experiências possam ser necessárias para centralizar exatamente o mostrador. Seis "vezes caracter" são selecionados para isso.

Registro 1

REG. #	ENDEREÇO A3 A0	FUNÇÃO		BIT	HEX.	DEC.
0	0000	CONTADOR DA LINHA HORIZ.	80	0 1 0 0 1 1 1 1	4F	79
1	0001	ENTRELAÇO LARGURA DO SINC. HORIZ. RETARDO DO SINC. HORIZ.	0 7 6	0 0 1 1 1 1 1 0	3E	62
2	0010	VARREDURA POR LINHA DE DADOS CARACTER POR LINHA	13 64	X 1 1 0 0 0 1 1	63	99
3	0011	ESPAÇO ENTRE CARACTERES LINHAS DE DADOS	1 16	1 0 0 0 1 1 1 1	8F	143
4	0100	VARREDURA POR QUADRO X =	262 3	0 0 0 0 0 0 1 1	03	3
5	0101	COMEÇO DO DADO VERTICAL 3 + RETARDO DA VARREDURA VERTICAL RETARDO DA VARREDURA	25 28	0 0 0 1 1 1 0 0	1C	28
6	0110	COMEÇO DO DADO ÚLTIMA LINHA DE DADOS MOSTRADA (= LINHAS DE DADO)		X X 0 0 1 1 1 1	0F	15

Tabela 9.2 Folha de operação de programação dos registros do CRT 5027 para um formato de tela 16 X 64.

Registro 2: Este possui dois campos:

1) Bits de 3 a 6 (bit 7 não é usado) possuem o número de varreduras por caracter. Neste caso, nós definimos o caracter como 10 X 13, então é utilizado o binário equivalente de 13 − 1 = 12 (todos os contadores do CRT 5027 começam em 0, não em 1, então a programação dos contadores será sempre menos 1 do que o número).

2) Bits de 0 a 2 contém um código de 3 bits para o número de caracteres por linha. Das folhas de dados do componente temos que o código para 64 é 011.

Registro 2

Registro 3: Este possui dois campos:

1) Bits 6 e 7 retardam o cursor e a sincronização para permitir os retardos de propagação do gerador de caracteres e da memória programável.

2) Bits de 0 a 5 definem o número de linhas de dados, começando com o binário zero para uma linha. Será programado como 16 − 1 = 15.

Registro 3

Registro 4: O registro 4 possui o número de linhas rastreadas por quadro. Para o modo não entrelace este número deriva da fórmula (N − 256) : 2 = 3.

Registro 4

Registro 5: Este contém o número de linhas rastreadas entre o início do pulso de sincronização vertical e o início do dado (sincronização vertical + parte não visível). Este tempo deve ser longo o suficiente para permitir o tempo de retraço completo do monitor e permitir o posicionamento vertical do display. Nós usaremos aqui 28. A parte visível será calculada pelo CRT 5027 como 262 − (13 X 16) − 28 = 26.

Registro 5

Registro 6: O registro 6 é o registro de rascunho, é programado com o número da linha de dado a ser mostrada. Quando queremos inicializar o CRT 5027, este será programado como o registro 3 (bits 6 e 7 não são utilizados).

Registro 6

Registro 7 e Registro 8: Estes registros possuem o número do caracter do cursor e o número da linha, respectivamente. Quando se quer posicionar inicialmente o cursor no canto esquerdo superior, ambos registros serão inicializados com zeros. A mudança subseqüente da posição do cursor será dada como descrito na "operação do circuito".

Registro 7
Registro 8

Descrição do circuito

Referindo-se à figura 9.2, os componentes CI1A, CI1B e CI4 fornecem o clock de ponto do vídeo (12,58 MHz) e o clock de caracter DCC, o qual é o clock de ponto: 10 (cada caracter tem 10 pontos de largura). O clock de ponto de vídeo determina a taxa real do dado de vídeo. O clock de caracter determina a velocidade que cada caracter é endereçado. O CI6A é um buffer de entrada do clock de ponto do CRT 8002. Um resistor de pull-up é usado na saída para garantir o nível lógico 1 da entrada VDC.

O comando LOAD carrega a informação de registro da PROM CI7 para o CRT 5027. A capacidade do CRT 5027 de seu autocarregar é utilizada para varrer automaticamente os endereços da PROM. O LOAD é gerado automaticamente pelo CI1D ao ligar-se o sistema.

Por causa da estrutura de via do CRT 5027, a informação da posição do cursor é carregada na mesma via como o registro de dado. Os selecionadores de dados, 3 estados, CI14 e CI15 selecionam o dado da posição X do cursor a partir do contador CI8 e CI7, ou o dado da posição Y do cursor a partir de CI1D. Os CI12 e CI13 selecionam o modo de endereçamento para o CRT 5027. Três modos são utilizados: "sem autocarregamento" para carga de registro, carrega posição X do cursor e carrega posição Y do cursor.

Os CI's de 16 a 21 decodificam o modo do atributo e controles do cursor da via de dados ASCII. Se atributos gráficos ou especiais não são desejados, os CI's 16, 17 e 21 não são necessários. Da mesma forma, se os controles de cursor forem possíveis diretamente, a decodificação destes não será necessária. Os CI's 19 e 20 são PROMs de 256 X 4. Suas programações são de acordo com as necessidades do usuário. A programação usada neste terminal está mostrada na tabela 9.3. Quando uma tecla designada como atributo ou tecla de modo for pressionada, a palavra de controle apropriada será colocada no CI21; todas as entradas de dados subseqüentes terão aquela palavra carregada nos 4 bits superiores da memória programável. Isto permite que o atributo ou o modo sejam mudados em uma base caracter por caracter. O CI18 é um decodificador de 2 para 4 e é habilitado quando um controle de cursor do tipo retorna à posição anterior, retorno de linha/salta linha, ou ↑ for decodificado, fornecendo o movimento apropriado do cursor.

Pode-se usar tecnologia TTL ou TTL baixa potência, entretanto recomenda-se Shottky TTL para o CI6 devido ao rápido tempo de subida necessário para a entrada de clock.

Operação

Depois de ligar, deve-se apertar "CONTROL Q" para colocar o sistema no modo normal. Apertando a tecla SPACE e ERASE simultaneamente se limpará a tela. Todos os caracteres apertados serão mostrados normalmente. Se outros atributos ou gráficos são desejados, o código do controle apropriado deve ser entrado. Este caracter não será mostrado nem faz com que o cursor se mova na tela, mas entrará com um novo comando. Os modos podem ser trocados para cada caracter. O movimento do cursor pode ser decodificado da entrada ASCII pela tecla de controle como indicado na tabela 9.3.

Programação PROM

Teclado	Função	Endereço	PROM 1 Saída	PROM 2 Saída
		76543210	$D_1D_2D_3D_4$	$D_1D_2D_3D_4$
Return	Retorno do carro	00011011	0011	1000
LF	Alimentação de linha	00010101	1011	1000
Control H	Cursor à esquerda	00010001	0111	1000
RS	Cursor para cima	00111101	1111	1000
US	Cursor à direita	00111111	1111	1010
Control Q	Atributo normal	00100011	1111	1011
Control W	Pisca	00101111	1011	1011
Control E	Sublinha	00001011	0111	1011
Control R	Reverso	00100101	0011	1011
Control T	Modo externo	00101001	1101	1011
Control Y	Gráfico fino	00110011	1100	1011
Control U	Gráfico largo	00101011	1110	1011
PROM			0011	1110

Tabela 9.3 Programação da PROM para o circuito da figura 9.2.

NÚMERO	TIPO
CI1	74LS04
CI2	CRT8002
CI3	CRT5207
CI4	74LS160
CI5	74LS20
CI6	74LS04
CI7	HM7603 PROM
CI8	74193
CI9	74193
CI10	74193
CI11	7408
CI12	74LS02
CI13	74LS74
CI14	74LS257
CI15	74LS257
CI16	74LS02
CI17	74LS00
CI18	74155
CI19	HM7621A PROM1
CI20	HM7621 PROM2
CI21	74174
CI22–CI33	2102A-4

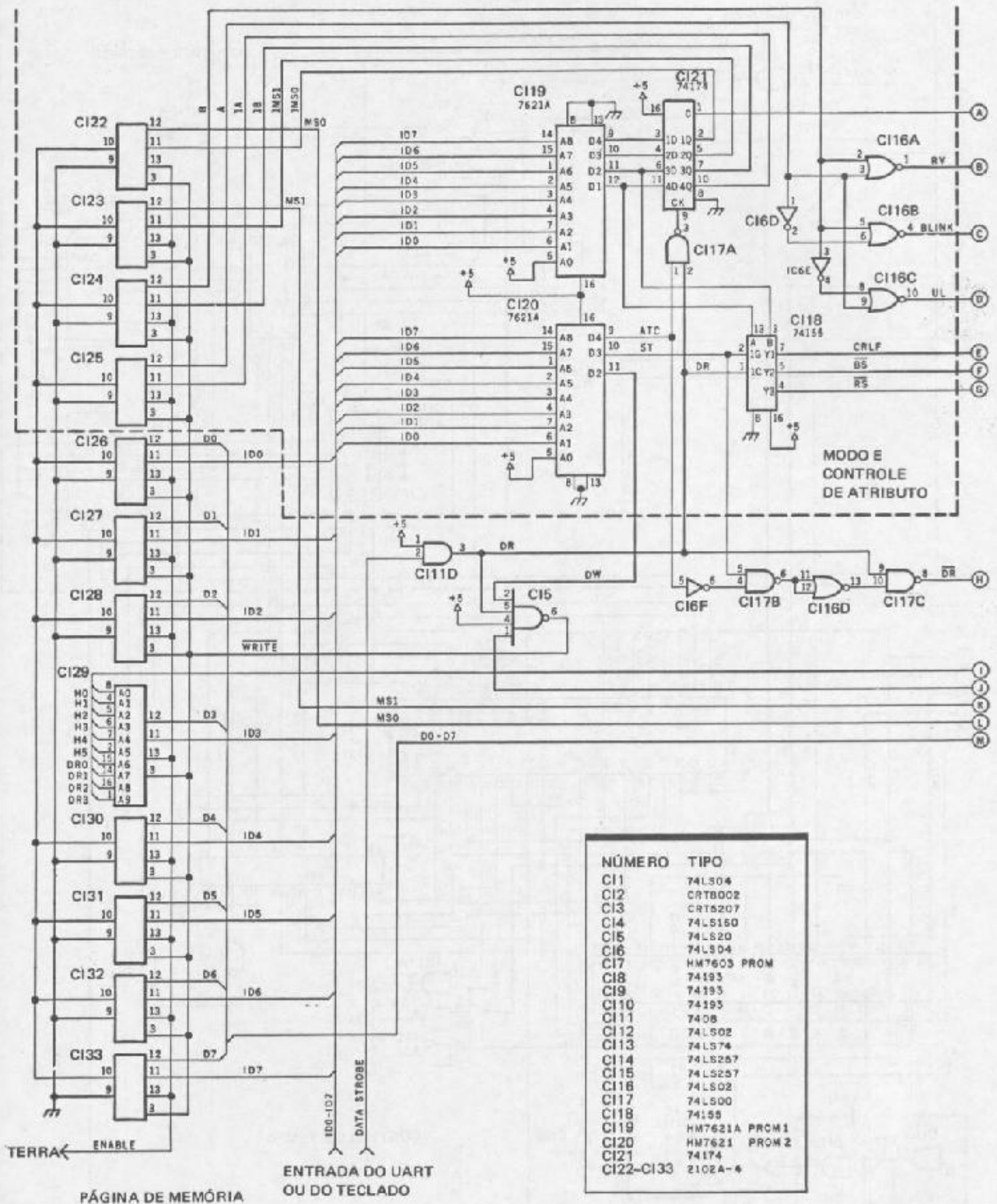



Figura 9.2 Diagrama esquemático de um terminal de baixo custo usando o CI 5027 e CI 8002.

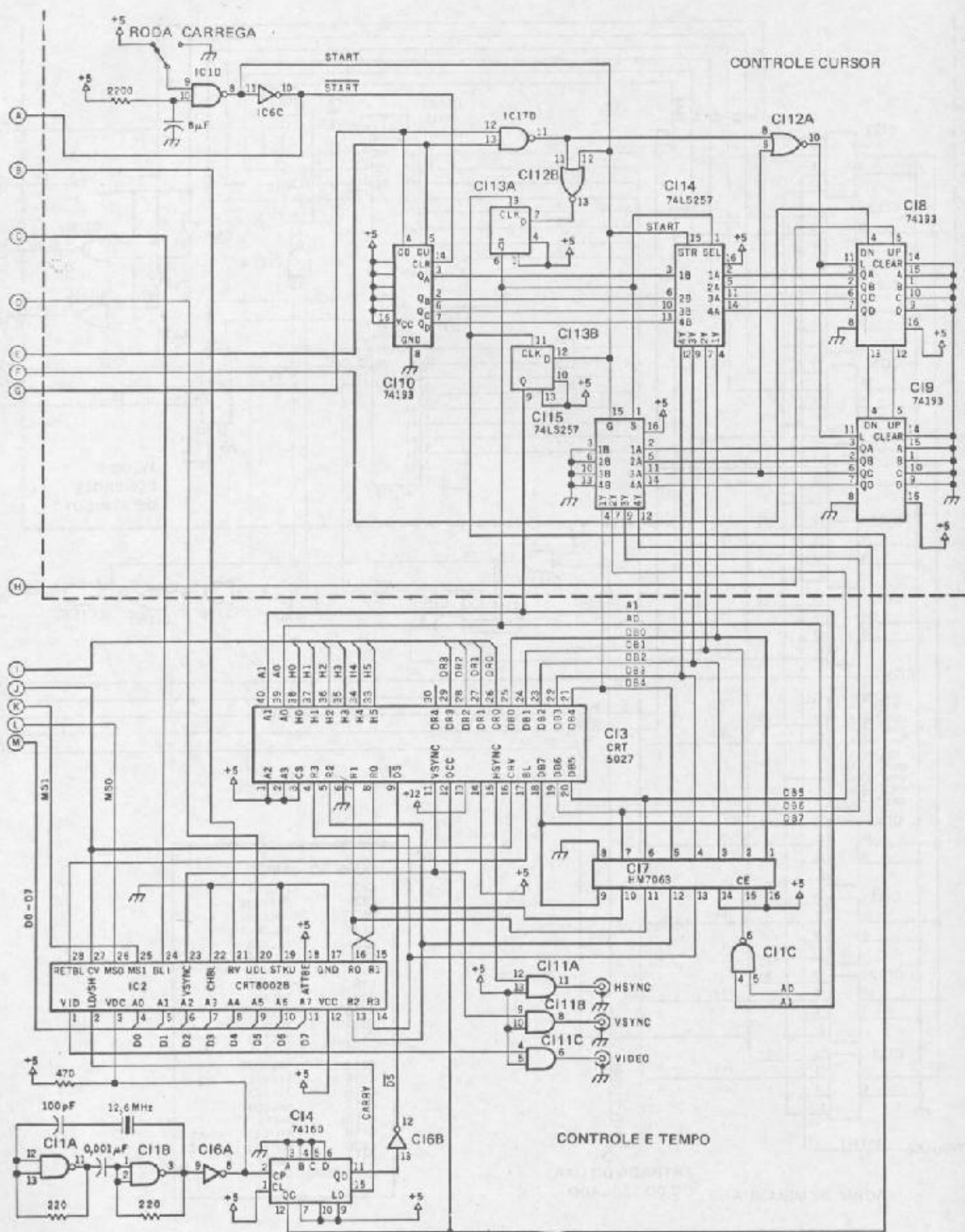



Figura 9.2 Continuação

O resto do sistema

A figura 9.3 mostra o circuito necessário para se ter uma interface RS232-C. O uso de integrados MOS de larga escala de integração reduz o número de integrados a um mínimo.

Um codificador para teclado do tipo KR2376 (CI1) codifica e retira o bounce (centelhamento da chave) das chaves e entrega o caracter codificado em ASCII para o COM 2017 UART (veja apêndices C6 e C7). O UART em troca faz a interface serial. A razão da freqüência dos dados é programável através das chaves na entrada do COM 8046 que é um gerador de razão de dados (veja apêndice C10).

Variações do terminal

O terminal descrito pode facilmente ser modificado para uma grande variedade de outros formatos de tela. As seguintes mudanças são necessárias para 80 caracteres por linha para 24 linhas.

1) Varredura horizontal – 312 linhas, freqüência horizontal de 20.220 Hz.

2) A freqüência do oscilador de vídeo é calculada como 9 (pontos por caracter) X 100 (caracteres por linha) X 20.220 = 19,198 MHz. Note que 9 pontos por caracter foi selecionado em vez de 10, com 10 teríamos uma freqüência de 20,2 MHz, que está além da máxima freqüência aceita pelo CRT 8002A. O CI4 deve ser preparado para dividir por 9 em vez de 10.

3) 1K byte adicional de memória de página é necessário. A figura 9.4 mostra as ligações de endereço necessárias.

4) A programação dos registros do CRT 5027 é mostrada nas tabelas 9.4 e 9.5.

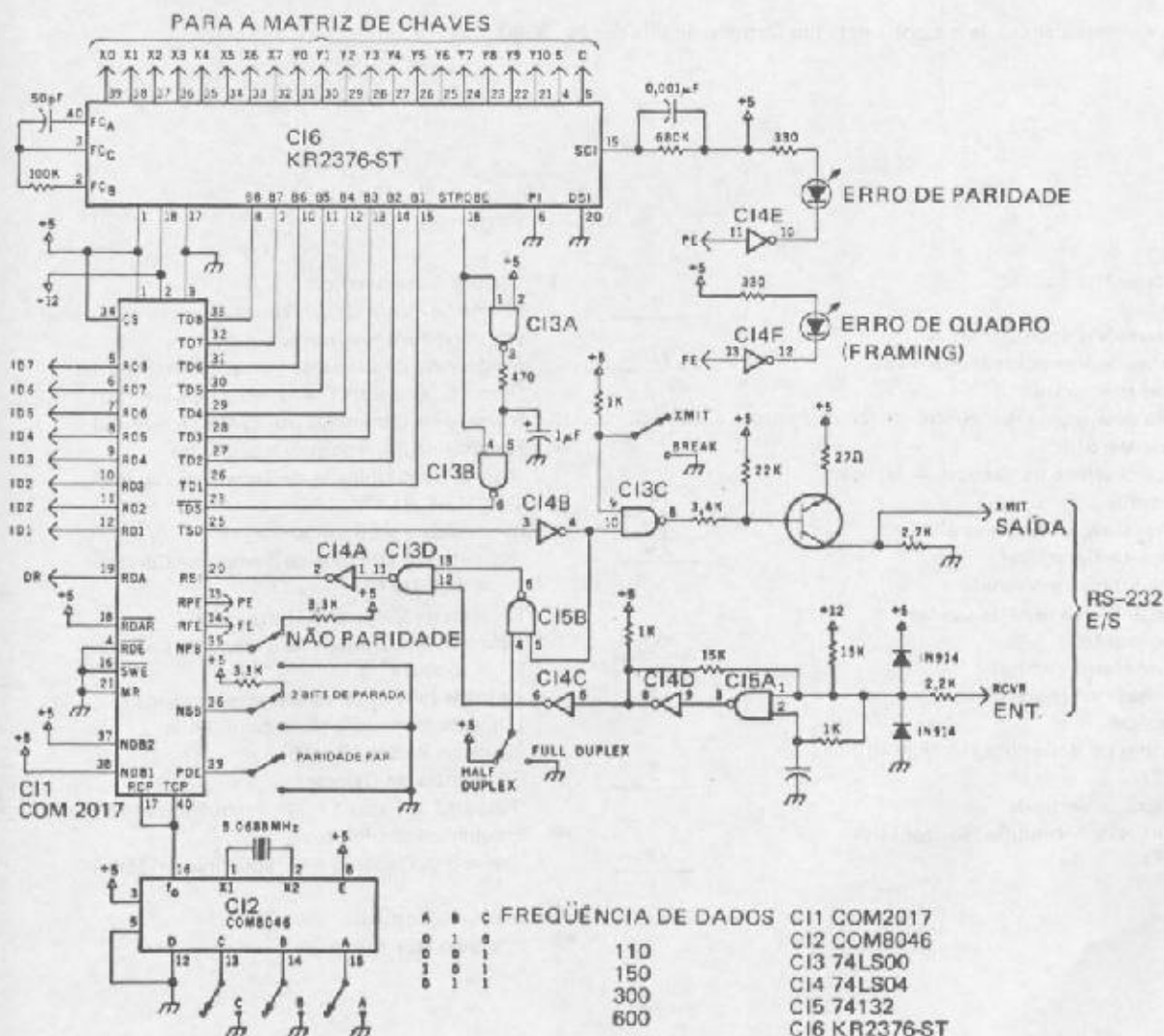



Figura 9.3 Diagrama esquemático de uma interface RS 232-C para um terminal de vídeo.

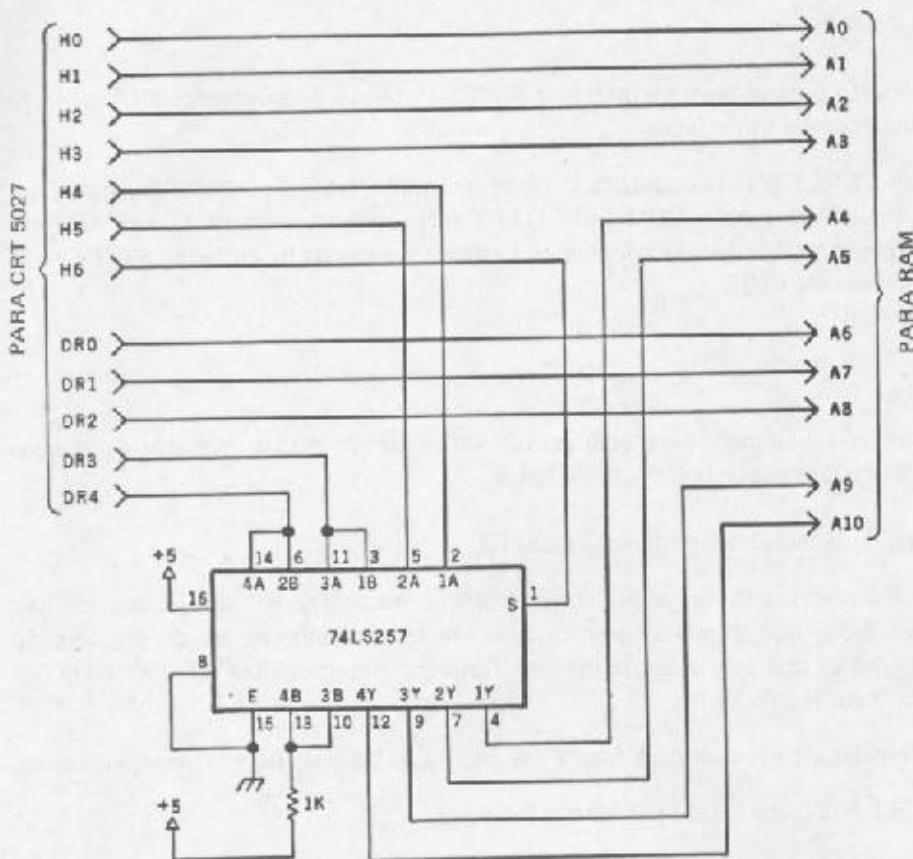



Figura 9.4 Sistema de mapeamento de memória para um formato de tela de 24 × 80.

1. Matriz de Caracteres Horizontais (Número de Pontos) — 7
2. Matriz de Caracteres Verticais (Número de Linhas da Varredura Horizontal) — 11
3. Bloco do Caracter Horizontal (Passo 1 + Espaço desejado na Horizontal = Nº de Pontos) — 9
4. Bloco do Caracter Vertical (Passo 2 + Espaço desejado na Vertical = Nº de Linhas de Varredura) — 13
5. Freqüência de Atualização de Quadro (Hz) — 60
6. Número de Linhas de Caracteres — 24
7. Número Total de Linhas de Varredura (Passo 4 × Passo 6 = Número de Linhas da Varredura Horizontal) — 312
8. Retardo do Sincronismo Vertical (Número em Linhas Horizontais) — 3
9. Sincronismo Vertical (Número em Linhas de Varredura Horizontal; T = 148,3 μs*) — 3
10. Retardo na Varredura Vertical (Número em Linhas de Varredura Horizontal; T = 890,2 μs*) — 18
11. Quadro Total Vertical (Some os Passos de 7 a 10 = Número em Linhas em Varredura Horizontal) — 336
12. Freqüência da Linha de Varredura Horizontal (Passo 5 × Passo 11 = Freqüência em Hz) — 20,220
13. Número de Caracteres por Linha Horizontal — 80
14. Retardo no Sincronismo Horizontal (Número em Unidade de Tempo de Caracter; T = 1,48 μs**) — 3
15. Sincronismo Horizontal (Número em Unidade de Tempo de Caracter; T = 4,94 μs**) — 10
16. Retardo na Varredura Horizontal (Número em Unidade de Tempo de Caracter; T = 3,46 μs**) — 7
17. Unidade de Tempo de Caracter em uma Linha de Varredura Horizontal Total (Some os Passos 13 a 16) — 100
18. Freqüência do Caracter (Passo 12 × Passo 17 = Freqüência em MHz) — 20220
19. Freqüência dos Pontos (Passo 3 × Passo 18 = Freqüência em MHz) — 18,198

* Intervalo Vertical
** Intervalo Horizontal

Tabela 9.4 Folha de dados do CRT 5027 para uma tela de formato não entrelaçado de 80 colunas por 24 linhas.

Nº REG.	ENDEREÇO	FUNÇÃO	BITS	HEX.	DEC.
0	0000	CONTADOR DA LINHA HOR. 100	0 1 1 0 0 0 1 1	63	99
1	0001	ENTRELAÇO 0 LARGURA DO SINCRONISMO HORIZONTAL 10 RETARDO DO SINC. HOR. 3	0 1 0 1 0 0 1 1	53	83
2	0010	VARREDURA POR LINHA DE DADOS 13 CARACTERES POR LINHA 80	X 1 1 0 0 1 0 1	65	101
3	0011	ESPAÇO ENTRE CARACTERES 2 LINHAS DE DADOS 24	1 0 0 1 0 1 1 1	97	151
4	0100	VARREDURA POR QUADRO 336 X = 40	0 0 1 0 1 0 0 0	28	40
5	0101	COMEÇO DO DADO VERTICAL = 3 + RETARDO DA VARREDURA VERTICAL: RETARDO DA VARREDURA 18 COMEÇO DO DADO 21	0 0 0 1 0 1 0 1	15	21
6	0110	ÚLTIMA LINHA DE DADOS MOSTRADOS (=LINHA DE DADOS)	X X 0 1 0 1 1 1	17	23

Tabela 9.5 Folha de dados para programação de um formato de tela de 24 X 80 do CRT 5027.

APÊNDICES

APÊNDICE A

TÉCNICAS DE CONSTRUÇÃO/MONTAGEM

TIPOS DE CONSTRUÇÃO

Como resultado de elaborar todo mês um projeto para minha coluna na revista "BYTE" chamado *CIARCIA'S CIRCUIT CELLAR* e de construir todos os circuitos deste livro, penso que posso lhes falar como uma autoridade no assunto de construção/montagem de protótipos. Protótipo é um bom termo para definir a primeira tentativa de montar-se um circuito, direto de um esquemático. É bem diferente de um kit ou de um projeto semimontado que já inclui uma placa de circuito impresso faltando somente inserir os componentes.

Montar o protótipo de um circuito não é fácil. Existem diversos problemas, portanto, montar um protótipo com sucesso é antes de tudo uma função da experiência de cada um, a qual só se adquire, construindo, projetando ou montando alguma coisa.

O texto foi escrito partindo desta filosofia. Eu sugiro que se comece pela fonte de alimentação, não só porque o resto do computador fica inoperante sem ela, mas também pelo fato da fonte possuir proteções internas que podem ser bem amenizantes no caso de se cometer algum erro. Também, construindo primeiro a fonte de alimentação, não se correrá o risco de danificar o resto do computador quando for testá-la.

Em geral, a principal regra ao se fazer um protótipo é ser meticuloso. O computador PAZ possui altas freqüências. A fiação entre duas conexões deve ser a mais curta possível. Quanto maior for uma fiação, maior efeito de antena ela produz (N.T. captando ruídos).

Em casos extremos, o computador pode realmente parar de funcionar devido à indução de ruídos elétricos. Para os sinais digitais (relativamente mais lentos), que são transportados pela fiação conectada às portas de E/S externas, esta situação fica menos crítica, mas pulsos estreitos e dados em alta velocidade, como os sinais de controle do processador central e as linhas de endereço, são mais críticos. Nestes casos, é sempre recomendável a utilização de circuitos adicionais de proteção como, por exemplo, os buffers. (N.T. Circuitos com maior capacidade de corrente.)

O computador PAZ pode ter o seu lay-out como vocês verão. A figura A1 sugere uma maneira típica: a montagem poderá ser com WIRE-WRAP (N.T. processo especial para montagem de protótipos onde o fio de wire-wrap é enrolado nos pinos dos soquetes, onde serão plugados os componentes) ou com solda. Qualquer placa, grande o suficiente para caber todos os CHIPS, deverá servir. Uma escolha que recomendo é a placa standard para protótipos S-100, encontrada em muitas lojas de computadores. Não existe nenhuma outra barra particular além das normais para os sinais do Z80, designados para o PAZ, porque ele é um sistema projetado para ocupar apenas uma placa. O

conector de 100 pinos proporciona uma conexão conveniente para E/S e para alimentação. Caso se decida dividir o esquemático do computador e utilizar mais de uma placa para a montagem, dever-se-á tomar muito cuidado. A divisão deverá ser feita apenas entre subsistemas lógicos; para maior sucesso, todos os sinais deverão ser buferizados (N.T. neologismo proveniente da palavra buffer, que significa – ampliação da capacidade de corrente) na entrada e saídas das placas e toda a memória poderá ser montada num cartão separado.

Como já dito anteriormente, as linhas de dados e endereços já estão devidamente buferizadas.

Figura A.1 Um lay-out típico para o computador PAZ.

A questão da montagem à solda ou com WIRE-WRAP é uma prerrogativa do construtor. Pessoalmente, prefiro a soldagem ponto a ponto porque é mais fácil de modificar na fase de depuração. O WIRE-WRAP talvez seja melhor quando o circuito do PAZ já tenha sido testado e melhorado.

Uma fiação para a fonte de alimentação longa e serial (passando por cada chip) deve ser evitada. Melhor do que utilizar apenas um longo fio para +5V e outro para terra é utilizar uma placa de dupla face e assim empregar a face superior para ligar +5V e a inferior para o terra. Com este enfoque, cada CHIP poderá ser plugado (utilizando-se soquetes apropriados) e os pinos de alimentação podem ser soldados diretamente nas trilhas de cobre da placa. Utilizando-se ou não o WIRE-WRAP, é recomendável soldar os pinos de alimentação para reduzir o potencial de conexões intermitentes (maus contatos). Utilizar o plano de terra para fazer a fiação é uma das melhores maneiras de reduzirmos ruídos em computadores. Caso não possuam um plano de terra, soldem então um fio grosso em torno do perímetro da placa e levem extensões curtas até ele.

Outra coisa essencial para se fazer um protótipo de computador são os capacitores de desacoplamento. Os circuitos integrados digitais, na maioria das aplicações, trabalham em condições de esforço e sem cuidados com a temperatura sendo, portanto, sucetíveis a ruídos provenientes da linha de alimentação. É comum entrarem até em oscilação devido a este ruído. O problema pode ser eliminado colocando-se um capacitor de 0,01 μF até 0,1 μF entre +5V e terra de três em três CI's. Outro recurso adicional é colocar um capacitor eletrolítico nas entradas de cada conexão de fontes de alimentação CC na placa. Geralmente estes capacitores são de tântalo e para as três fontes do PAZ serão necessários três capacitores.

Finalmente, se vocês gostaram do conceito de projeto do PAZ, mas preferem dedicar mais tempo utilizando o produto final, do que testar suas técnicas de construção e montagem, vocês podem procurar comprar as EPROMs programadas para o monitor do PAZ.

APÊNDICE B

CÓDIGOS ASCII

Dec	Octal	Hex	Paridade	Caracter	Teclado	Nome dos códigos
000	000	00	Even	NUL	@	NULL, CTRL SHIFT P, TAPE LEADER
001	001	01	Odd	SOH	A	START OF HEADER, SOM
002	002	02	Odd	STX	B	START OF TEXT, EOA
003	003	03	Even	ETX	C	END OF TEXT, EOM
004	004	04	Odd	EOT	D	END OF TRANSMISSION, END
005	005	05	Even	ENQ	E	ENQUIRY, WRU, WHO ARE YOU
006	006	06	Even	ACK	F	ACKNOWLEDGE, RU, ARE YOU
007	007	07	Odd	BEL	G	BELL
008	010	08	Odd	BS	H	BACKSPACE, FE0
009	011	09	Even	HT	I	HORIZONTAL TAB, TAB
010	012	0A	Even	LF	J	LINE FEED, NEW LINE, NL
011	013	0B	Odd	VT	K	VERTICAL TAB, VTAB
012	014	0C	Even	FF	L	FORM FEED, FORM, PAGE
013	015	0D	Odd	CR	M	CARRIAGE RETURN, EOL
014	016	0E	Odd	SO	N	SHIFT OUT, RED SHIFT
015	017	0F	Even	SI	O	SHIFT IN, BLACK SHIFT
016	020	10	Odd	DLE	P	DATA LINK ESCAPE, DC0
017	021	11	Even	DC1	Q	XON, READER ON
018	022	12	Even	DC2	R	TAPE, PUNCH ON
019	023	13	Odd	DC3	S	XOFF, READER OFF
020	024	14	Even	DC4	T	TAPE, PUNCH OFF
021	025	15	Odd	NAK	U	NEGATIVE ACKNOWLEDGE, ERR
022	026	16	Odd	SYN	V	SYNCHRONOUS IDLE, SYNC
023	027	17	Even	ETB	W	END OF TEXT BUFFER, LEM
024	030	18	Even	CAN	X	CANCEL, CANCL
025	031	19	Odd	EM	Y	END OF MEDIUM
026	032	1A	Odd	SUB	Z	SUBSTITUTE
027	033	1B	Even	ESC	[	ESCAPE, PREFIX
028	034	1C	Odd	FS	\	FILE SEPARATOR
029	035	1D	Even	GS	]	GROUP SEPARATOR
030	036	1E	Even	RS	^	RECORD SEPARATOR
031	037	1F	Odd	US	–	UNIT SEPARATOR
032	040	20	Odd	SP		SPACE, BLANK

033	041	21	Even	!	
034	042	22	Even	"	
035	043	23	Odd	#	
036	044	24	Even	$	
037	045	25	Odd	%	
038	046	26	Odd	&	
039	047	27	Even	'	APOSTROPHE
040	050	28	Even	(	
041	051	29	Odd	)	
042	052	2A	Odd	*	
043	053	2B	Even	+	
044	054	2C	Odd	,	COMMA
045	055	2D	Even	–	MINUS
046	056	2E	Even	.	
047	057	2F	Odd	/	
048	060	30	Even	0	NUMBER ZERO
049	061	31	Odd	1	NUMBER ONE
050	062	32	Odd	2	
051	063	33	Even	3	
052	064	34	Odd	4	
053	065	35	Even	5	
054	066	36	Even	6	
055	067	37	Odd	7	
056	070	38	Odd	8	
057	071	39	Even	9	
058	072	3A	Even	:	
059	073	3B	Odd	;	
060	074	3C	Even	<	LESS THAN
061	075	3D	Odd	=	
062	076	3E	Odd	>	GREATER THAN
063	077	3F	Even	?	
064	100	40	Odd	@	SHIFT P
065	101	41	Even	A	
066	102	42	Even	B	
067	103	43	Odd	C	
068	104	44	Even	D	
069	105	45	Odd	E	
070	106	46	Odd	F	
071	107	47	Even	G	
072	110	48	Even	H	
073	111	49	Odd	I	LETTER I
074	112	4A	Odd	J	
075	113	4B	Even	K	
076	114	4C	Odd	L	
077	115	4D	Even	M	
078	116	4E	Even	N	
079	117	4F	Odd	O	LETTER O
080	120	50	Even	P	
081	121	51	Odd	Q	
082	122	52	Odd	R	
083	123	53	Even	S	
084	124	54	Odd	T	
085	125	55	Even	U	
086	126	56	Even	V	
087	127	57	Odd	W	
088	130	58	Odd	X	
089	131	59	Even	Y	
090	132	5A	Even	Z	
091	133	5B	Odd	[	SHIFT K
092	134	5C	Even	\	SHIFT L
093	135	5D	Odd	]	SHIFT M
094	136	5E	Odd	^	↑, SHIFT N
095	137	5F	Even	_	←, SHIFT O, UNDERSCORE
096	140	60	Even	`	ACCENT GRAVE
097	141	61	Odd	a	
098	142	62	Odd	b	
099	143	63	Even	c	

100	144	64	Odd	d	
101	145	65	Even	e	
102	146	66	Even	f	
103	147	67	Odd	g	
104	150	68	Odd	h	
105	151	69	Even	i	
106	152	6A	Even	j	
107	153	6B	Odd	k	
108	154	6C	Even	l	
109	155	6D	Odd	m	
110	156	6E	Odd	n	
111	157	6F	Even	o	
112	160	70	Odd	p	
113	161	71	Even	q	
114	162	72	Even	r	
115	163	73	Odd	s	
116	164	74	Even	t	
117	165	75	Odd	u	
118	166	76	Odd	v	
119	167	77	Even	w	
120	170	78	Even	x	
121	171	79	Odd	y	
122	172	7A	Odd	z	
123	173	7B	Even	{	
124	174	7C	Odd	|	VERTICAL SLASH
125	175	7D	Even	}	ALTMODE
126	176	7E	Even	~	(ALTMODE)
127	177	7F	Odd	DEL	DELETE, RUBOUT

APÊNDICE C

FOLHAS DE ESPECIFICAÇÃO DO FABRICANTE

APÊNDICE C1

2708

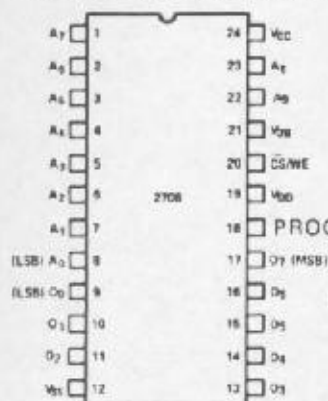
8K(1K × 8) UV Erasable Prom (Prom Apagável por UV)

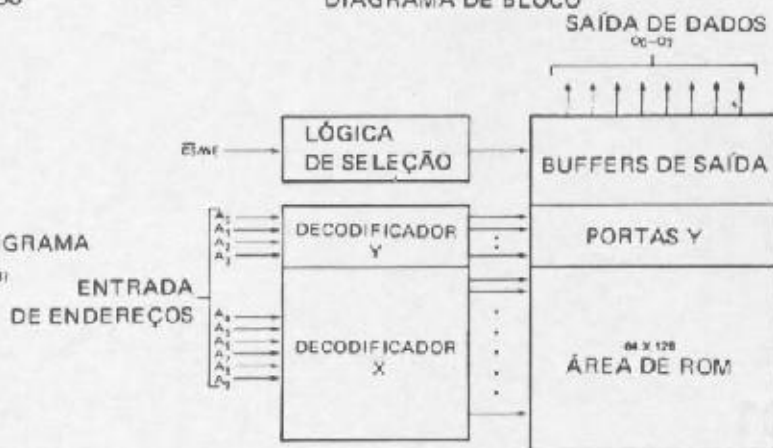
	Potência Máx.	Acesso Máx.
2708	800mW	450ns
2708L	425mW	450ns
2708-1	800mW	350ns
2708-6	800mW	550ns

- **Baixa dissipação de potência – 425 mW Máx. (2708L)**
- **Tempo de acesso rápido – 350 ns Máx. (2708-1)**
- **Estática – não necessita clock**
- **Entrada e saída de dados compatíveis com TTL durante os modos de programação e leitura**
- **Saídas Tri-State-Multiconectáveis**

A 2708 da Intel é uma EPROM de 8192 bits apagável por raios ultravioleta e reprogramável eletricamente, utilizada especialmente quando se necessitam rápidas alterações de programas e experimentos de diferentes padrões de gravação. Todas as entradas e saídas de dados são compatíveis aos níveis TTL durante os modos de programação e de leitura. As saídas são TRI-STATE (N.T. ficam em alta impedância enquanto não ativadas), permitindo um interfaceamento direto com as barras do sistema.

A 2708L com 425 mW foi desenhada para sistemas que necessitem uma menor dissipação de potência do que a obtida com a 2708. Uma diminuição de 50% de dissipação de potência, sem nenhuma perda de velocidade, pode ser obtida com a 2708L. A 2708L possui alta imunidade de ruído de entrada e 10% de tolerância nas tensões de alimentação. Para microprocessadores que necessitem de tempo de acesso rápido, a 2708-1 de alta velocidade com 350 ns está disponível.

A família 2708 é fabricada com portas de silicone canal-N de tecnologia FAMOS e é encontrada em encapsulamento de 24 pinos DUAL-IN-LINE (N.T. pinos paralelos nos 2 lados).

NOME DOS PINOS

A_0-A_9	ADDRESS INPUTS
O_1-O_8	DATA OUTPUTS/INPUTS
$\overline{CS}$/WE	CHIP SELECT/WRITE ENABLE INPUT

CONEXÃO DOS PINOS DURANTE LEITURA DE PROGRAMAÇÃO

MODE	PIN NUMBER							
	DATA I/O 9-11, 13-17	ADDRESS INPUTS 1-8, 22, 23	V_{SS} 12	PROGRAM 18	V_{DD} 19	$\overline{CS}$/WE 20	V_{BB} 21	V_{CC} 24
READ	D_{OUT}	A_{IN}	GND	GND	+12	V_{IL}	-5	+5
DESELECT	HIGH IMPEDANCE	DON'T CARE	GND	GND	+12	V_{IH}	-5	+5
PROGRAM	D_{IN}	A_{IN}	GND	PULSED 26V	+12	V_{IHW}	-5	+5

FAMÍLIA 2708

PROGRAMAÇÃO

As especificações para programação estão descritas na seção de instruções para programação do catálogo de dados PROM/ROM.

Absolute Maximum Ratings (Valores Máximos Absolutos)

Temperature Under Bias .... -25°C to +85°C
Storage Temperature .... -65°C to +125°C
V_{DD} With Respect to V_{BB} .... +20V to -0.3V
V_{CC} and V_{SS} With Respect to V_{BB} .... +15V to -0.3V
All Input or Output Voltages With Respect to V_{BB} During Read .... +15V to -0.3V
$\overline{CS}$/WE Input With Respect to V_{BB} During Programming .... +20V to -0.3V
Program Input With Respect to V_{BB} .... +35V to -0.3V
Power Dissipation 1.5W

* COMENTÁRIO
Esforços maiores do que os especificados nos "Valores Máximos Absolutos" (Absolute Max. Ratings) podem danificar permanentemente o componente. Estes são apenas valores máximos de esforço e a operação funcional do componente nestas condições não está prevista. A exposição aos valores máximos absolutos por longos períodos afetam a confiabilidade do componente.

CONDIÇÕES DE OPERAÇÃO CC e CA DURANTE A LEITURA

	2708	2708-1	2708-6	2708L
Temperature Range	0°C-70°C	0°C-70°C	0°C-70°C	0°C-70°C
V_{CC} Power Supply	5V ± 5%	5V ± 5%	5V ± 5%	5V ± 10%
V_{DD} Power Supply	12V ± 5%	12V ± 5%	12V ± 5%	12V ± 10%
V_{BB} Power Supply	−5V ± 5%	−5V ± 5%	−5V ± 5%	−5V ± 10%

OPERAÇÃO DE LEITURA

CARACTERÍSTICAS CC E DE OPERAÇÃO

Symbol	Parameter	2708, 2708-1, 2708-6 Limits			2708L Limits			Units	Test Conditions
		Min.	Typ.[2]	Max.	Min.	Typ.[2]	Max.		
I_{LI}	Address and Chip Select Input Sink Current		1	10		1	10	μA	V_{IN} = 5.25V or V_{IN} = V_{IL}
I_{LO}	Output Leakage Current		1	10		1	10	μA	V_{OUT} = 5.5V, CS/WE = 5V
I_{DD}[3]	V_{DD} Supply Current		50	65		21	25	mA	Worst Case Supply Currents[4]
I_{CC}[3]	V_{CC} Supply Current		6	10		2	4	mA	All Inputs High;
I_{BB}[3]	V_{BB} Supply Current		30	45		10	14	mA	CS/WE = 5V; T_A = 0°C
V_{IL}	Input Low Voltage	V_{SS}		0.65	V_{SS}		0.65	V	
V_{IH}	Input High Voltage	3.0		V_{CC}+1	2.2		V_{CC}+1	V	
V_{OL}	Output Low Voltage			0.45			0.4	V	I_{OL} = 1.6mA (2708, 2708-1, 2708-6) I_{OL} = 2mA (2708L)
V_{OH1}	Output High Voltage	3.7			3.7			V	I_{OH} = −100μA
V_{OH2}	Output High Voltage	2.4			2.4			V	I_{OH} = −1mA
PD	Power Dissipation			800			325	mW	T_A = 70°C
							425	mW	T_A = 0°C

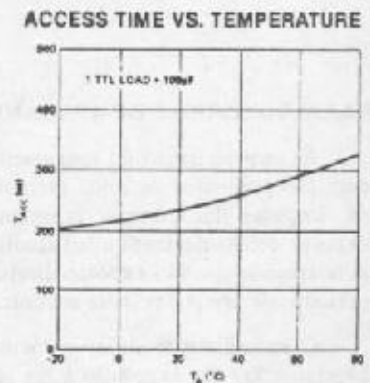
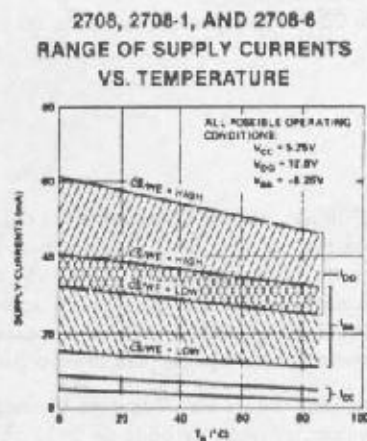
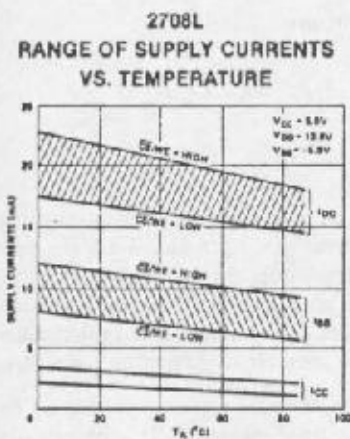
NOTAS:

1. V_{BB} deve ser aplicada antes de V_{CC} e V_{DD}. V_{BB} também deve ser a última tensão a ser desligada.
2. Valores típicos são para $T_A = 25°C$ e tensões nominais.
3. A dissipação de potência total não é calculada somando-se as várias correntes (I_{DD}, I_{CC}, I_{BB}) e multiplicando-se pelas respectivas voltagens, porque existe circulação de corrente entre as várias fontes e V_{SS}. As correntes I_{DD}, I_{CC} e I_{BB} podem ser utilizadas somente para determinar a capacidade da fonte de alimentação.
4. Para a 2708L, I_{BB} é especificada para o estado programado e é de 18 mA máx. no estado não programado.

FAMÍLIA 2708

2708L RANGE OF SUPPLY CURRENTS VS. TEMPERATURE

2708, 2708-1, AND 2708-6 RANGE OF SUPPLY CURRENTS VS. TEMPERATURE

ACCESS TIME VS. TEMPERATURE

CARACTERÍSTICAS CA

Symbol	Parameter	2708, 2708L Limits		2708-1 Limits		2708-6 Limits		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
t_{ACC}	Address to Output Delay		450		350		550	ns
t_{CO}	Chip Select to Output Delay		120		120		160	ns
t_{DF}	Chip Deselect to Output Float	0	120	0	120	0	160	ns
t_{OH}	Address to Output Hold	0		0		0		ns

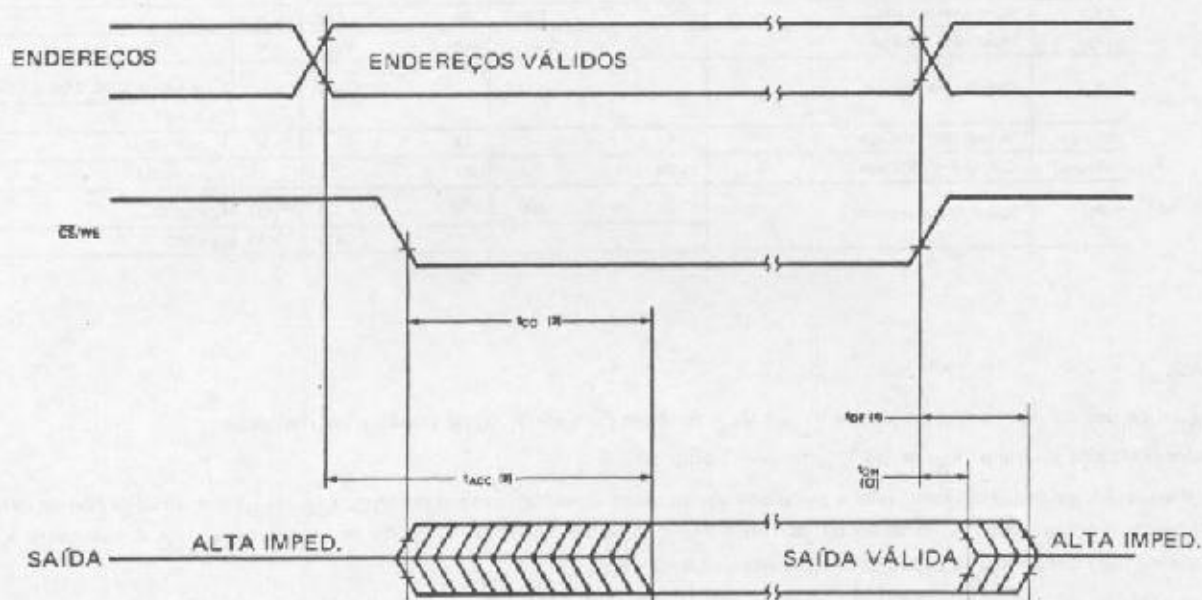
CAPACITÂNCIA[1] $T_A = 25°C$, $f = 1$ MHz

Symbol	Parameter	Typ.	Max.	Unit.	Conditions
C_{IN}	Input Capacitance	4	6	pF	$V_{IN} = 0V$
C_{OUT}	Output Capacitance	8	12	pF	$V_{OUT} = 0V$

NOTA: 1. Este parâmetro é amostrado periodicamente e não é testado a 100%.

CONDIÇÕES DE TESTE CA

Output Load: 1 TTL gate and $C_L = 100$ pF
Input Rise and Fall Times: ≤20 ns
Timing Measurement Reference Levels: 0.8V and 2.8V for inputs: 0.8V and 2.4V for outputs.
Input Pulse Levels: 0.65V to 3.0V

FORMAS DE ONDA AC[2]

NOTAS:

2. Todos os tempos nos parênteses são mínimos e estão em nseg, a menos que especificado em contrário.
3. $\overline{CS}$ deve ter um atraso de $t_{ACC} - t_{CO}$ após os endereços estarem válidos sem impacto para t_{ACC}.
4. t_{DF} é especificado a partir de $\overline{CS}$ ou da troca de endereços, ou o que ocorrer primeiro.

CARACTERÍSTICAS DE APAGAMENTO

As características do apagamento da família 2708 são tais que o apagamento começa a acontecer quando se expõe à luz ultravioleta com comprimentos de onda menores do que aproximadamente 4000 Ångstroms (Å). Deve-se observar que a luz do sol e certo tipo de lâmpadas fluorescentes possuem comprimentos de onda entre 3000-4000 Å. Os dados mostram que um componente típico, exposto constantemente à luz fluorescente de um quarto, pode ser apagado em aproximadamente 3 anos, enquanto levaria 1 semana para apagá-lo quando exposto diretamente à luz do sol. Se a 2708 tiver de ser exposta a estas condições de iluminação por prolongados períodos de tempo, pode-se encontrar através da Intel etiquetas opacas para tapar o visor da 2708, para prevenir apagamentos acidentais.

O procedimento de apagamento recomendado para a família 2708 (ver Catálogo de Dados PROM/ROM seção de instruções de programação) é a exposição à luz ultravioleta com comprimento de onda de 2537 Ångstroms (Å). A dose integrada (i.e., intensidade UV X tempo de exposição) para apagamento deve ser de no mínimo $15W\text{-seg}/cm^2$. O tempo de apagamento com esta dosagem é de aproximadamente 15 a 20 minutos usando uma lâmpada ultravioleta com uma potência efetiva de $12000MW/cm^2$. O componente deve ser colocado a uma distância de 2,5 cm da lâmpada durante o apagamento. Algumas lâmpadas possuem um filtro em seus tubos que deve ser removido antes do apagamento.

APÊNDICE C2

2716

16K (2K × 8) UV Erasable Prom (Prom Apagável por UV)

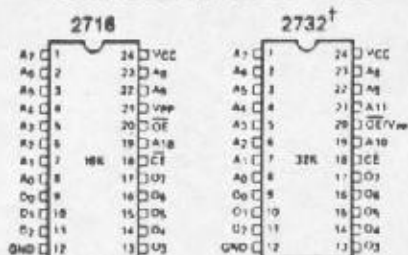
- **Tempo de acesso rápido**
  - **350 ns Máx. 2716-1**
  - **390 ns Máx. 2716-2**
  - **450 ns Máx. 2716**
  - **490 ns Máx. 2716-5**
  - **650 ns Máx. 2716-6**
- **Fonte única +5V**
- **Baixa dissipação de potência**
  - **525 mW Máx. potência ativa**
  - **132 mW Máx. potência inativa (Standby)**
- **Pinagem compatível com a EPROM 2732-INTEL®**
- **Necessidades simples para programação**
  - **programação de uma posição isolada**
  - **programação com pulso de 50 ms**
- **Entradas e saídas compatíveis com TTL durante leitura e programação**
- **Totalmente estática**

A 2716 da INTEL é uma EPROM de 16.384 bits apagável por raios ultravioleta e reprogramável eletricamente. A 2716 opera com apenas uma fonte de 5V, possui um modo estático de Standby (espera) e uma programação de posições isoladas. Isto torna o projeto com EPROMs mais rápido, mais fácil e mais econômico.

A 2716 com sua única fonte de 5V e com tempo de acesso de 350 ns torna-se ideal para utilização com os novos microprocessadores de alto desempenho com fonte +5V, tais como os 8085 e 8086 da Intel. As 2716-5 e 2716-6 estão disponíveis para aplicações de velocidades mais baixas.

A 2716 é também a primeira EPROM com um modo de espera estática (Standby) e que reduz a dissipação de potência sem aumentar o tempo de acesso. A dissipação máxima de potência ativa é de 525 mW, enquanto a dissipação máxima de potência inativa (Standby) é de apenas 132 mW, ou seja, 75% de economia.

A 2716 possui o mais simples e o mais rápido método já visto para programação de EPROMs – programação por um único pulso TTL. Não são necessários pulsos de alta voltagem porque todos os controles de programação são executados por sinais TTL. Programa qualquer posição a qualquer instante, seja isoladamente, seqüencialmente ou aleatoriamente. O tempo total de programação para os 16.384 bits é de 100 segundos.

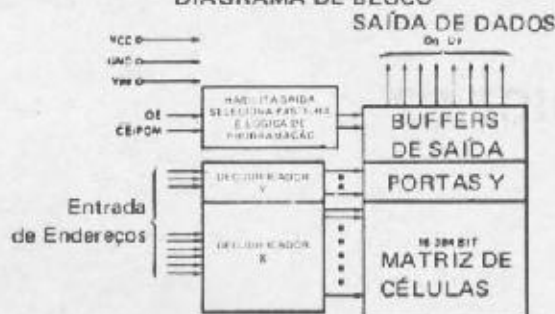
CONFIGURAÇÃO DOS PINOS

Veja folha de dados da 2732 para especificações

NOME DOS PINOS

A_0–A_{10}	ADDRESSES
$\overline{CE}$/PGM	CHIP ENABLE/PROGRAM
$\overline{OE}$	OUTPUT ENABLE
O_0–O_7	OUTPUTS

MODOS DE SELEÇÃO

MODE \ PINS	$\overline{CE}$/PGM (18)	$\overline{OE}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Standby	V_{IH}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IH}	+25	+5	High Z

DIAGRAMA DE BLOCO

PROGRAMAÇÃO

As especificações para programação estão descritas na seção de instruções para programação do catálogo de dados PROM/ROM.

Absolute Maximum Ratings (Valores Máximos Absolutos)

Temperature Under Bias . . . . −10°C to +80°C
Storage Temperature . . . . −65°C to +125°C
All Input or Output Voltages with
Respect to Ground . . . . +6V to −0.3V
V_{PP} Supply Voltage with Respect
to Ground During Program . . . . +26.5V to −0.3V

* COMENTÁRIO

Esforços maiores do que os especificados nos "Valores Máximos Absolutos" (Absolute Max. Ratings) podem danificar permanentemente o componente. Estes são apenas valores máximos de esforço e a operação funcional do componente nestas condições não está prevista. A exposição aos valores máximos absolutos por longos períodos afetam a confiabilidade do componente.

Condições de Operação CC e CA durante a Leitura

	2716	2716-1	2716-2	2716-5	2716-6
Temperature Range	0°C – 70°C	0°C – 70°C	0°C – 70°C	0°C – 70°C	0°C – 70°C
V_{CC} Power Supply [1,2]	5V ±5%	5V ±10%	5V ±5%	5V ±5%	5V ±5%
V_{PP} Power Supply [2]	V_{CC}	V_{CC}	V_{CC}	V_{CC}	V_{CC}

OPERAÇÃO DE LEITURA

Características CC e de Operação

Symbol	Parameter	Limits Min.	Limits Typ. [3]	Limits Max.	Unit	Conditions
I_{LI}	Input Load Current			10	μA	V_{IN} = 5.25V
I_{LO}	Output Leakage Current			10	μA	V_{OUT} = 5.25V
I_{PP1} [2]	V_{PP} Current			5	mA	V_{PP} = 5.25V
I_{CC1} [2]	V_{CC} Current (Standby)		10	25	mA	$\overline{CE}$ = V_{IH}, $\overline{OE}$ = V_{IL}
I_{CC2} [2]	V_{CC} Current (Active)		57	100	mA	$\overline{OE}$ = $\overline{CE}$ = V_{IL}
V_{IL}	Input Low Voltage	−0.1		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}+1	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 2.1 mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = −400 μA

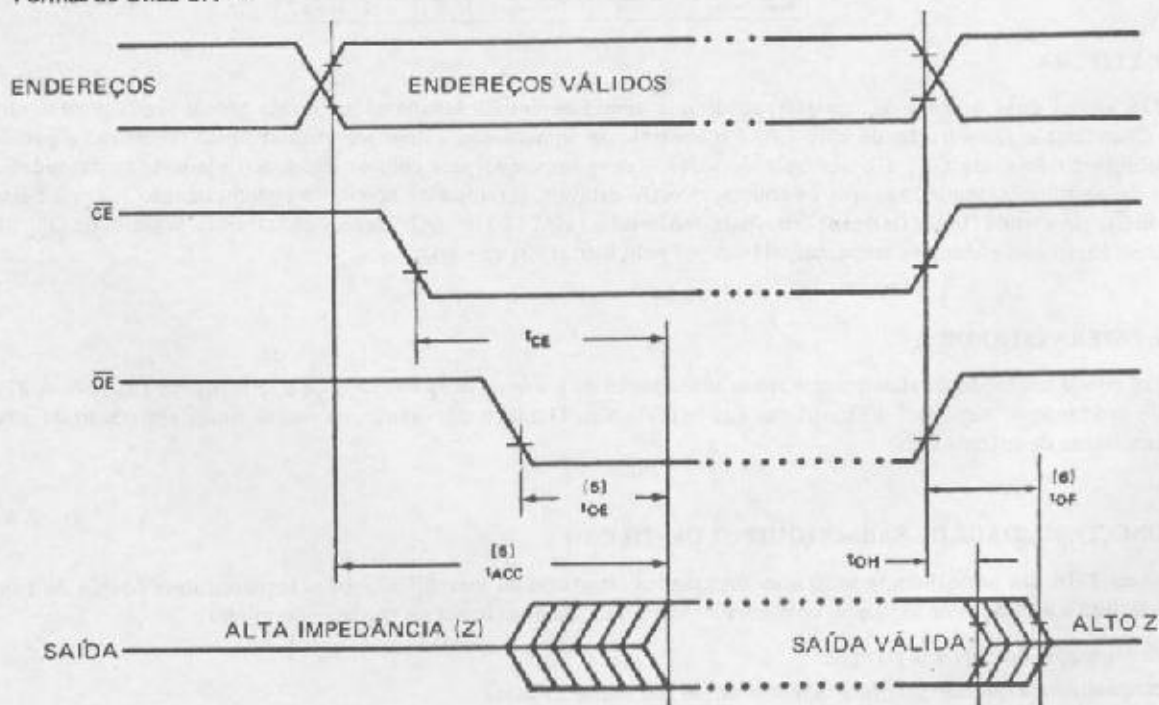
NOTAS:

1. V_{CC} deve ser aplicada ao mesmo tempo ou antes de V_{PP} e desligada ao mesmo tempo ou depois de V_{PP}.
2. V_{PP} deve ser conectada diretamente a V_{CC} exceto durante a programação. A corrente fornecida será então a soma de I_{CC} e I_{PP1}.
3. Valores típicos para $T_A = 25°C$ e tensões nominais.
4. Este parâmetro é apenas amostrado e não testado a 100%.

Características Típicas

Características CA

Symbol	Parameter	Limits (ns) 2716 Min.	2716 Max.	2716-1 Min.	2716-1 Max.	2716-2 Min.	2716-2 Max.	2716-5 Min.	2716-5 Max.	2716-6 Min.	2716-6 Max.	Test Conditions
t_{ACC}	Address to Output Delay		450		350		390		450		450	$\overline{CE} = \overline{OE} = V_{IL}$
t_{CE}	$\overline{CE}$ to Output Delay		450		350		390		490		650	$\overline{OE} = V_{IL}$
t_{OE}	Output Enable to Output Delay		120		120		120		160		200	$\overline{CE} = V_{IL}$
t_{DF}	Output Enable High to Output Float	0	100	0	100	0	100	0	100	0	100	$\overline{CE} = V_{IL}$
t_{OH}	Output Hold from Addresses, $\overline{CE}$ or $\overline{OE}$ Whichever Occurred First	0		0		0		0		0		$\overline{CE} = \overline{OE} = V_{IL}$

Formas de Onda CA [1]

Capacitância [4] $T_A = 25°C$, f = 1 MHz

Symbol	Parameter	Typ.	Max.	Unit	Conditions
C_{IN}	Input Capacitance	4	6	pF	$V_{IN} = 0V$
C_{OUT}	Output Capacitance	8	12	pF	$V_{OUT} = 0V$

Condições de Teste CA

Output Load: 1 TTL gate and $C_L = 100$ pF
Input Rise and Fall Times: ≤20 ns
Input Pulse Levels: 0.8V to 2.2V
Timing Measurement Reference Level:
Inputs 1V and 2V
Outputs 0.8V and 2V

NOTAS:

1. V_{CC} deve ser aplicada ao mesmo tempo ou antes de V_{PP} e desligada ao mesmo tempo ou depois de V_{PP}.
2. V_{PP} deve ser conectada diretamente a V_{CC} exceto durante a programação. A corrente fornecida será, então, a soma de I_{CC} e I_{PP1}.
3. Valores típicos para $T_A = 25°C$ e tensões nominais.
4. Este parâmetro é apenas amostrado e não testado a 100%.
5. OE deve ter um retardo de $T_{ACC} - t_{DE}$ após a transição negativa de CE sem impactar t_{ACC}.
6. t_{DF} está especificado a partir de OE ou CE, ou o que ocorrer primeiro.

CARACTERÍSTICAS DE APAGAMENTO

As características de apagamento da 2716 são tais que o apagamento começa a acontecer quando se expõe à luz com comprimentos de onda menores do que aproximadamente 4000 Ångstroms (Å). Deve-se observar que a luz do sol e certo tipo de lâmpada fluorescente possuem comprimentos de onda entre 3000–4000 Å. Os dados mostram que um componente típico exposto constantemente à luz fluorescente de um quarto pode ser apagado em aproximadamente 3 anos, enquanto levaria apenas 1 semana para apagá-lo quando exposto a estas condições de iluminação por prolongados períodos de tempo, poderão ser encontradas, através da Intel, etiquetas opacas para tapar o visor da 2716 para prevenir apagamentos acidentais.

O procedimento de apagamento recomendado para a família 2716 é a exposição à luz ultravioleta com comprimento de onda de 2537 Ångstroms (Å). A dose integrada (i.e., intensidade UV × tempo de exposição) para apagamento deve ser de no mínimo 15W-seg/cm². O tempo de apagamento com esta dosagem é de aproximadamente 15 a 20 minutos, usando uma lâmpada ultravioleta com uma potência efetiva de 12000 μW/cm². O componente deve ser colocado a uma distância de 2,5 cm da lâmpada durante o apagamento. Algumas lâmpadas possuem um filtro em seus tubos que deve ser removido antes do apagamento.

OPERAÇÃO DO COMPONENTE/PASTILHA

Os cinco modos de operação da 2716 estão mostrados na tabela 1. Deve-se notar que todas as entradas para os cinco modos são de níveis TTL. As fontes de alimentação requeridas são as de +5V e a V_{PP}. A fonte V_{PP} deve estar em 25V durante os três modos de programação e deve estar em 5V nos outros dois modos.

TABELA 1 – SELEÇÃO DOS MODOS

MODE \ PINS	$\overline{CE}$/PGM (18)	$\overline{OE}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Standby	V_{IH}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IH}	+25	+5	High Z

MODO DE LEITURA

A 2716 possui duas funções de controle, sendo que ambas devem ser satisfeitas na devida ordem lógica para se obter dados nas saídas. O Chip-Enable (habilitação do chip $\overline{CE}$) é o controle de alimentação e deve ser utilizado para selecionar a pastilha. O Output Enable (habilitação de saída $\overline{OE}$) é o controle de saída e deve ser usado para colocar dados nos pinos de saída, independentemente da seleção da pastilha. Assumindo-se que os endereços estão estáveis, o tempo de acesso de endereçamento (t_{ACC}) é igual ao retardo existente da $\overline{CE}$ até a saída (t_{CE}). Os dados estarão disponíveis na saída 120 ns (t_{OE}) depois da transição negativa de $\overline{OE}$, pressupondo-se que $\overline{CE}$ esteve baixo e os endereços estiveram estáveis por pelo menos $t_{ACC} - t_{OE}$.

MODO DE ESPERA (STANDBY)

A 2716 possui um modo de standby que reduz a dissipação de potência ativa em 75%, de 525 mW para 132 mW. A 2716 é colocada neste modo aplicando-se um nível TTL alto na sua entrada $\overline{CE}$. Durante este modo, as saídas ficam em estado de alta impedância, independentemente da entrada $\overline{OE}$.

MULTICONECTABILIDADE DE SAÍDAS (OUTPUT OR-TIEING)

Como as 2716 são normalmente utilizadas em grandes estruturas de memórias, a Intel forneceu uma função de controle de duas linhas que facilita a utilização de múltiplas conexões de memórias. As duas linhas de controle permitem:

a) A menor dissipação de potência possível.

b) A segurança absoluta que não ocorrerá uma contenção nas barras de saída.

Para uma utilização mais eficiente destas duas últimas linhas, é recomendado que o sinal $\overline{CE}$ (pino 18) seja decodificado e utilizado como uma função de seleção primária da pastilha, enquanto $\overline{OE}$ (pino 20) seja interligado entre todas as pastilhas da estrutura de memória e conectado ao sinal READ da barra de controle do sistema. Isto assegurará que todas as memórias não selecionadas estarão no modo de standby (baixo consumo) e que os pinos da saída estarão ativos apenas quando se desejar algum dado de uma pastilha qualquer.

PROGRAMAÇÃO

Inicialmente, e após cada apagamento, todos os bits da 2716 estarão no estado "1". Os dados serão introduzidos, programando-se "0's" seletivos nos bits desejados. Embora apenas os "0's" sejam efetivamente programados (gravados), os dados de programação podem conter ambos estados "1" e "0". A única maneira de se trocar um "0" por um "1" é por apagamento com ultravioleta.

A 2716 entra em modo de programação quando a tensão de V_{PP} está com 25V e $\overline{OE}$ está com V_{IH}. Os dados a serem programados são aplicados nos 8 bits em paralelo dos pinos de saída. Os níveis necessários para as entradas de dados e endereços são TTL.

Quando os endereços e os dados estão estáveis, um pulso TTL, ativo alto, de 50 mseg é aplicado na entrada $\overline{CE}$/PGM. Um pulso de programação deve ser aplicado para cada posição de memória a ser programada. Pode-se programar qualquer posição a qualquer instante – seja individual (isoladamente), seqüencial ou aleatoriamente. O pulso de programação deve ter uma largura máxima de 55 mseg. A 2716 não deve ser programada com um nível CC aplicado à entrada $\overline{CE}$/PGM.

A programação das múltiplas 2716 em paralelo com o mesmo dado pode ser facilmente realizada, devido à simplicidade das necessidades para programação. As entradas equivalentes das 2716 em paralelo podem ser interligadas quando forem programadas com o mesmo dado. Um pulso TTL, ativo alto, aplicado na entrada $\overline{CE}$/PGM, programa as 2716 em paralelo.

INIBIÇÃO DE PROGRAMAÇÃO

A programação das múltiplas 2716 em paralelo com dados diferentes pode também ser facilmente realizada. Com exceção da entrada $\overline{CE}$/PGM, todos os outros sinais equivalentes (incluindo $\overline{OE}$) podem ser interligados. Um pulso TTL, ativo alto, aplicado à entrada $\overline{CE}$/PGM de uma 2716 particular com V_{PP} em 25V, irá programar apenas aquela 2716. Um nível baixo na entrada $\overline{CE}$/PGM inibe a programação das outras 2716.

VERIFICAÇÃO DA PROGRAMAÇÃO

Uma verificação deverá ser feita nos bits programados para determinar se a gravação foi feita corretamente. A verificação pode ser feita com V_{PP} em 25V; exceto durante a programação e a verificação, V_{PP} deverá estar com 5V.

APÊNDICE C3

2102A, 2102AL/8102A-4*
1K × 1 BIT STATIC RAM (RAM ESTÁTICA)

P/N	Standby Pwr. (mW)	Operating Pwr. (mW)	Access (ns)
2102AL-4	35	174	450
2102AL	35	174	350
2102AL-2	42	342	250
2102A-2	——	342	250
2102A	——	289	350
2102A-4	——	289	450

- **Fonte única +5V**
- **Diretamente compatível com TTL: todas as entradas e saídas**
- **Modo de Standby (espera com baixo consumo)**
- **Saídas Tri-State: capacidade de multiconexão**
- **Entradas protegidas: todas as entradas com proteção contra carga estática**
- **Encapsulamento de baixo custo: configuração Dual-In-Line de 16 pinos**

A 2102A da Intel é uma Ram Estática de alta velocidade com 1024 bits que utiliza dispositivos MOS Canal-N integrados numa rede monolítica. Ela utiliza circuitos estáveis CC (estáticos), não necessitando, portanto, de relógios (clocks) ou de restauração (Refresh) para funcionar. O dado de saída é lido sem ser destruído e tem a mesma polaridade do dado de entrada.

A 2102A foi projetada para aplicações que têm como objetivos importantes o baixo custo, o alto desempenho, grande armazenamento de bits e simples interfaceamento. *Uma versão de baixo consumo por modo de standby (2102AL) também é encontrada. Ela possui exatamente as mesmas características da operação da 2102A, incluindo a facilidade de consumir potência de 35 mW em Standby e 174 mW em operação.*

É completamente compatível com TTL: entradas, saídas e fonte única de +5V. Um pino separado para o chip enable ($\overline{CE}$) permite uma fácil seleção de pastilhas individuais quando as saídas são multiconectadas.

A 2102A da Intel é fabricada com tecnologia de silício Canal-N. Esta tecnologia proporciona o projeto e a produção de circuitos MOS de alta performance e de utilização simples e fornece uma alta densidade de integração numa pastilha monolítica que pode utilizar tanto a tecnologia MOS convencional quanto a de silício Canal-P.

FAMÍLIA 2102 A

Absolute Maximum Ratings* (Valores Máximos Absolutos)

Ambient Temperature Under Bias	−10°C to 80°C
Storage Temperature	−65°C to +150°C
Voltage On Any Pin With Respect To Ground	−0.5V to +7V
Power Dissipation	1 Watt

* COMENTÁRIO

Esforços maiores do que os especificados nos "Valores Máximos Absolutos" (Absolute Max. Ratings) podem danificar permanentemente o componente. Estes são apenas valores máximos de esforço, e a operação funcional do componente nestas condições não está prevista. A exposição aos valores máximos absolutos por longos períodos afetam a confiabilidade do componente.

Características CC e de Operação

T_A = 0°C a 70°C, V_{CC} = 5V ±5% a menos que especificado em contrário

Symbol	Parameter	2102A, 2102A-4 2102AL, 2102AL-4 Limits			2102A-2, 2102AL-2 Limits			Unit	Test Conditions
		Min.	Typ.[1]	Max.	Min.	Typ.[1]	Max.		
I_{LI}	Input Load Current		1	10		1	10	μA	V_{IN} = 0 to 5.25V
I_{LOH}	Output Leakage Current		1	5		1	5	μA	$\overline{CE}$ = 2.0V, V_{OUT} = V_{OH}
I_{LOL}	Output Leakage Current		-1	-10		-1	-10	μA	$\overline{CE}$ = 2.0V, V_{OUT} = 0.4V
I_{CC}	Power Supply Current		33	Note 2		45	65	mA	All Inputs = 5.25V, Data Out Open, T_A = 0°C
V_{IL}	Input Low Voltage	-0.5		0.8	-0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4			0.4	V	I_{OL} = 2.1mA
V_{OH}	Output High Voltage	2.4			2.4			V	I_{OH} = -100μA

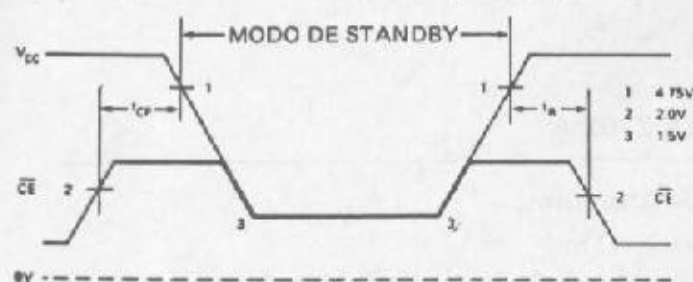
NOTAS:

1. Valores típicos para $T_A = 25°C$ e tensões nominais.
2. O valor máximo de I_{CC} é de 55 mA para 2102A e 2102A-4; 33 mA para 2102AL e 2102AL-4.

Características de Standby 2102AL, 2102AL-2, e 2102A-4 (Disponíveis apenas com Encapsulamento Plástico)

T_A = 0°C to 70°C

Symbol	Parameter	2102AL, 2102AL-4 Limits Min.	Typ.[1]	Max.	2102AL-2 Limits Min.	Typ.[1]	Max.	Unit	Test Conditions
V_{PD}	V_{CC} in Standby	1.5			1.5			V	
V_{CES}[2]	$\overline{CE}$ Bias in Standby	2.0			2.0			V	$2.0V \leq V_{PD} \leq V_{CC}$ Max.
		V_{PD}			V_{PD}			V	$1.5V \leq V_{PD} < 2.0V$
I_{PD1}	Standby Current		15	23		20	28	mA	All Inputs = V_{PD1} = 1.5V
I_{PD2}	Standby Current		20	30		25	38	mA	All Inputs = V_{PD2} = 2.0V
t_{CP}	Chip Deselect to Standby Time	0			0			ns	
t_R[3]	Standby Recovery Time	t_{RC}			t_{RC}			ns	

FORMAS DE ONDA EM STANDBY (ESPERA)

NOTAS:

1. Valores típicos para T_A = 25°C.
2. Considere as seguintes condições de teste: se a tensão de STANDBY (V_{PD}) estiver entre 5,25V (V_{CC}Máx.) e 2,0V, então $\overline{CE}$ deve ter nível de pelo menos 2,0V (V_{IH}). Se a tensão de STANDBY é menor que 2,0V, mas maior que 1,5V (V_{PD} Mín.), então $\overline{CE}$ e a tensão de STANDBY devem estar no mínimo com o mesmo valor, ou se estiverem diferentes, $\overline{CE}$ deve ser o mais positivo dos dois.
3. $t_R = t_{RC}$ (tempo do Ciclo de Leitura)

Características CA T_A = 0°C a 70°C, V_{CC} = 5V ±5% a menos que especificado em contrário.

CICLO DE LEITURA

Symbol	Parameter	2102A-2, 2102AL-2 Limits (ns) Min.	Max.	2102A, 2102AL Limits (ns) Min.	Max.	2102A-4, 2102AL-4 Limits (ns) Min.	Max.
t_{RC}	Read Cycle	250		350		450	
t_A	Access Time		250		350		450
t_{CO}	Chip Enable to Output Time		130		180		230
t_{OH1}	Previous Read Data Valid with Respect to Address	40		40		40	
t_{OH2}	Previous Read Data Valid with Respect to Chip Enable	0		0		0	

CICLO DE ESCRITA

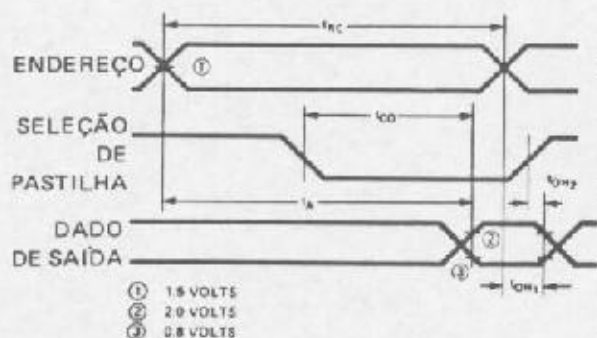
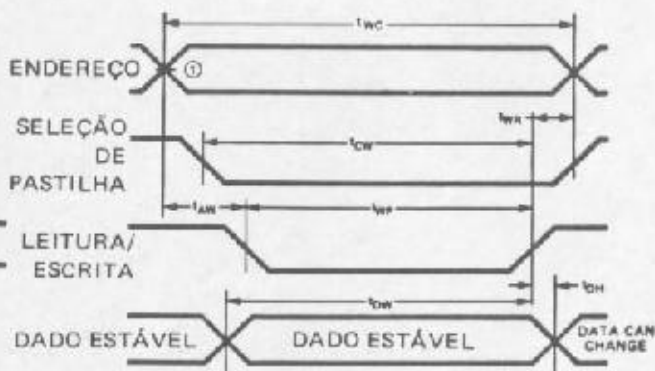
Symbol	Parameter	2102A-2, 2102AL-2 Min.	Max.	2102A, 2102AL Min.	Max.	2102A-4, 2102AL-4 Min.	Max.
t_{WC}	Write Cycle	250		350		450	
t_{AW}	Address to Write Setup Time	20		20		20	
t_{WP}	Write Pulse Width	180		250		300	
t_{WR}	Write Recovery Time	0		0		0	
t_{DW}	Data Setup Time	180		250		300	
t_{DH}	Data Hold Time	0		0		0	
t_{CW}	Chip Enable to Write Setup Time	180		250		300	

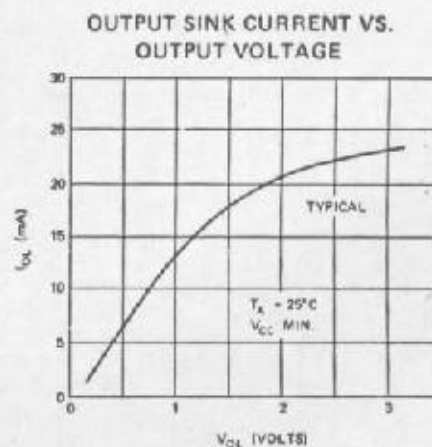
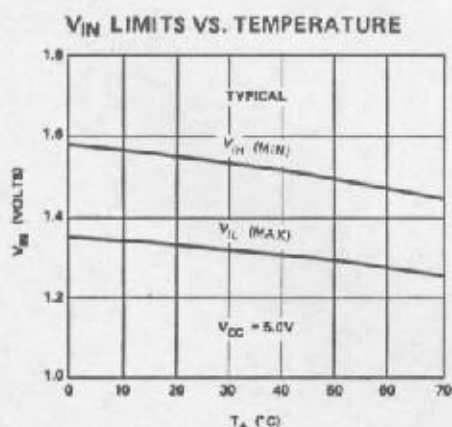
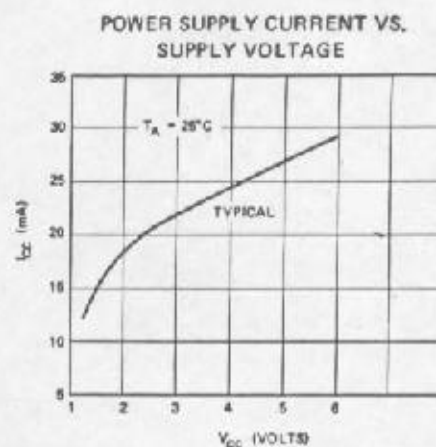
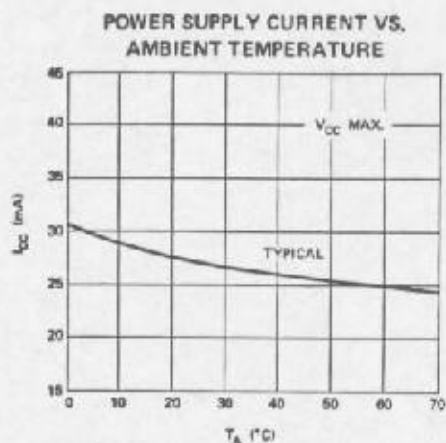
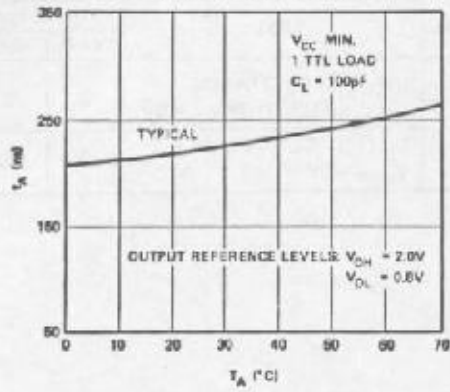
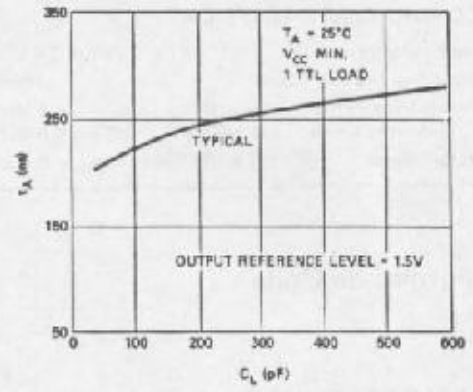
CONDIÇÕES DE TESTE CA

Input Pulse Levels:	0.8 Volt to 2.0 Volt
Input Rise and Fall Times:	10nsec
Timing Measurement Reference Levels — Inputs:	1.5 Volts
Output:	0.8 and 2.0 Volts
Output Load:	1 TTL Gate and C_L = 100 pF

CAPACITÂNCIA [2] T_A = 25°C, f = 1MHz

SYMBOL	TEST	LIMITS (pF)	
		TYP.[1]	MAX.
C_{IN}	INPUT CAPACITANCE (ALL INPUT PINS) V_{IN} = 0V	3	5
C_{OUT}	OUTPUT CAPACITANCE V_{OUT} = 0V	7	10

Formas de Onda

NOTAS: 1. Valores típicos para T_A = 25°C e tensões nominais.
2. Este parâmetro é amostrado periodicamente e não testado a 100%.

Características CC e CA típicas

APÊNDICE C4

2114A

1024 X 4 Bit Static Ram (Ram estática)

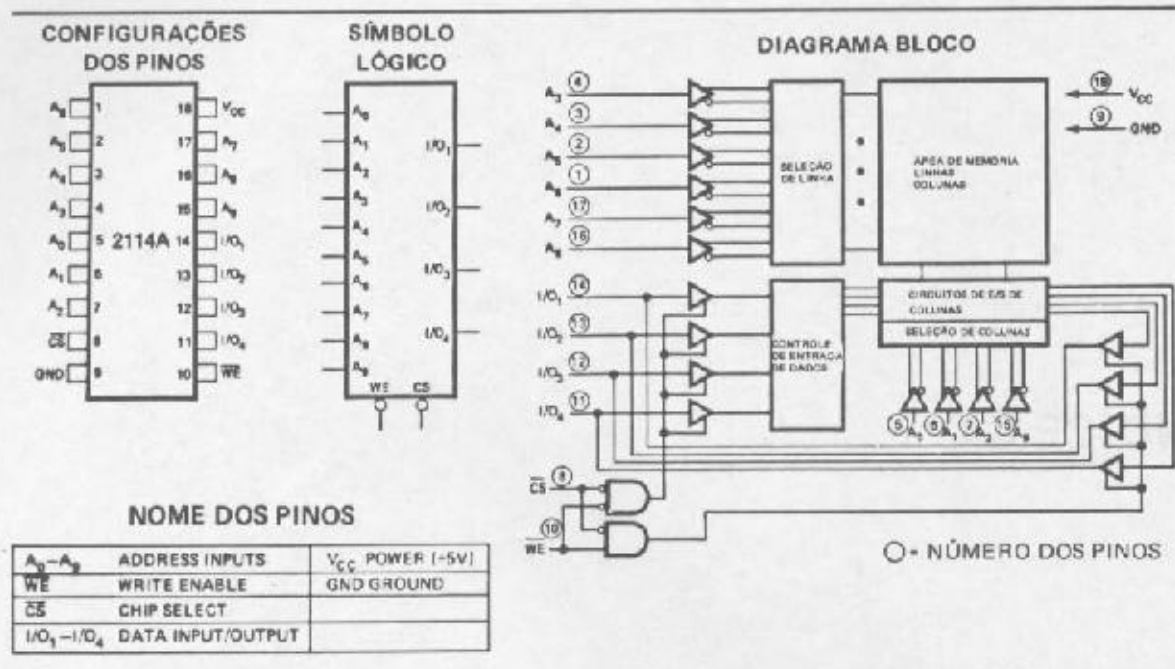
	2114AL-1	2114AL-2	2114AL-3	2114AL-4	2114A-4	2114A-5
Tempo de Acesso Máx. (ns)	100	120	150	200	200	250
Máxima Corrente (mA)	40	40	40	40	70	70

- Tecnologia HMOS
- Baixo consumo, alta velocidade
- Tempos de acesso e de ciclo idênticos
- Fonte única +5V ± 10%
- Encapsulamento de alta densidade com 18 pinos
- Memória totalmente estática - Não necessita de clocks ou de temporização
- Diretamente compatível com TTL: todas as entradas e saídas
- Dados comuns de entrada e saída, utilizando saídas Tri-State
- Aperfeiçoamento da 2114.

A 2114A da Intel é uma RAM de 4096 bits organizados como 1024 palavras de 4 bits que utiliza uma tecnologia HMOS de alto desempenho. Os circuitos utilizados internamente são todos estáveis CC (estáticos), não necessitando de relógios ou de restauração para funcionar. O acesso de dados é bem simples, devido à não-necessidade de estabilização do endereçamento. Os dados de saída lidos não são destruídos e possuem a mesma polaridade que os dados de entrada. Os pinos de entrada/saída de dados são comuns.

A 2114A foi projetada para aplicações de memória onde a alta performance e alta confiabilidade do HMOS, baixo custo, alta armazenagem de bits e interfaceamento simples são objetivos importantes. A 2114A é montada com encapsulamento de 18 pinos para uma mais alta densidade possível.

É diretamente compatível com TTL sob todos os aspectos: entradas, saídas e fonte única de +5V. Um sinal de CHIP SELECT ($\overline{CS}$), separado, permite uma fácil seleção individual de pastilhas multiconectadas.

A_0–A_9	ADDRESS INPUTS	V_{CC} POWER (+5V)
$\overline{WE}$	WRITE ENABLE	GND GROUND
$\overline{CS}$	CHIP SELECT	
I/O_1–I/O_4	DATA INPUT/OUTPUT	

FAMÍLIA 2114A

Absolute Maximum Ratings* (Valores Máximos Absolutos)

Temperature Under Bias .... −10°C to 80°C
Storage Temperature .... −65°C to 150°C
Voltage on any Pin
With Respect to Ground .... −3.5V to +7V
Power Dissipation 1.0W
D.C. Output Current 5mA

* COMENTÁRIO
Esforços maiores do que os especificados nos "Valores Máximos Absolutos" (Absolute Max. Ratings) podem danificar permanentemente o componente. Estes são apenas valores máximos de esforço e a operação funcional do componente nestas condições não está prevista. A exposição aos valores máximos absolutos por longos períodos afetam a confiabilidade do componente.

CARACTERÍSTICAS CC E DE OPERAÇÃO

$T_A = 0°C$ a $70°C$, $V_{CC} = 5V \pm 10\%$, a menos que especificado em contrário

SYMBOL	PARAMETER	2114AL-1/L-2/L-3/L-4 Min.	Typ.[1]	Max.	2114A-4/-5 Min.	Typ.[1]	Max.	UNIT	CONDITIONS
I_{LI}	Input Load Current (All Input Pins)			10			10	µA	V_{IN} = 0 to 5.5V
I_{LO}	I/O Leakage Current			10			10	µA	$\overline{CS} = V_{IH}$, $V_{I/O}$ = GND to VCC
I_{CC}	Power Supply Current		25	40		50	70	mA	V_{CC} = max, $I_{I/O}$ = 0 mA, $T_A = 0°C$
V_{IL}	Input Low Voltage	−3.0		0.8	−3.0		0.8	V	
V_{IH}	Input High Voltage	2.0		6.0	2.0		6.0	V	
I_{OL}	Output Low Current	2.1	9.0		2.1	9.0		mA	V_{OL} = 0.4V
I_{OH}	Output High Current	−1.0	−2.5		−1.0	−2.5		mA	V_{OH} = 2.4V
I_{OS}[2]	Output Short Circuit Current			40			40	mA	

NOTAS:
1. Valores típicos para T_A = 25°C e tensões nominais.
2. A duração não deve exceder 30 segundos.

Capacitância

$T_A = 25°C$, f = 1.0 MHz

SYMBOL	TEST	MAX	UNIT	CONDITIONS
$C_{I/O}$	Input/Output Capacitance	5	pF	$V_{I/O}$ = OV
C_{IN}	Input Capacitance	5	pF	V_{IN} = OV

Nota: Este parâmetro é periodicamente amostrado e não 100% testado.

CONDIÇÕES DE TESTE CA

Input Pulse Levels .... 0.8 Volt to 2.0 Volt
Input Rise and Fall Times .... 10 nsec
Input and Output Timing Levels .... 1.5 Volts
Output Load .... 1 TTL Gate and C_L = 100 pF

CARACTERÍSTICAS AC T_A = 0°C a 70°C, V_{CC} = 5V ± 10%, a menos que especificado em contrário

CICLO DE LEITURA[1]

SYMBOL	PARAMETER	2114AL-1 Min.	2114AL-1 Max.	2114AL-2 Min.	2114AL-2 Max.	2114AL-3 Min.	2114AL-3 Max.	2114A-4/L-4 Min.	2114A-4/L-4 Max.	2114A-5 Min.	2114A-5 Max.	UNIT
t_{RC}	Read Cycle Time	100		120		150		200		250		ns
t_A	Access Time		100		120		150		200		250	ns
t_{CO}	Chip Selection to Output Valid		70		70		70		70		85	ns
t_{CX}	Chip Selection to Output Active	10		10		10		10		10		ns
t_{OTD}*	Output 3-state from Deselection		30		35		40		50		60	ns
t_{OHA}	Output Hold from Address Change	15		15		15		15		15		ns

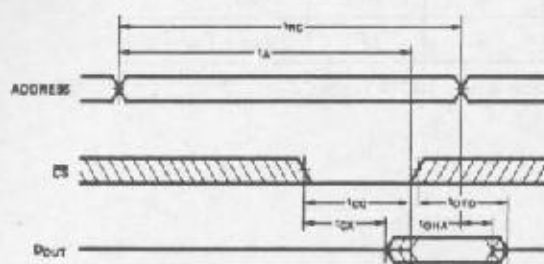
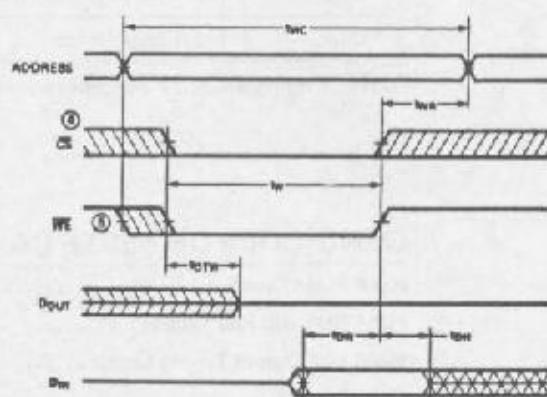
CICLO DE ESCRITA[2]

SYMBOL	PARAMETER	2114AL-1 Min.	2114AL-1 Max.	2114AL-2 Min.	2114AL-2 Max.	2114AL-3 Min.	2114AL-3 Max.	2114A-4/L-4 Min.	2114A-4/L-4 Max.	2114A-5 Min.	2114A-5 Max.	UNIT
t_{WC}	Write Cycle Time	100		120		150		200		250		ns
t_W	Write Time	75		75		90		120		135		ns
t_{WR}	Write Release Time	0		0		0		0		0		ns
t_{OTW}	Output 3-state from Write		30		35		40		50		60	ns
t_{DW}	Data to Write Time Overlap	70		70		90		120		135		ns
t_{DH}	Data Hold from Write Time	0		0		0		0		0		ns

NOTAS:

1. Uma leitura ocorre durante a superposição de um $\overline{CS}$ baixo a um $\overline{WE}$ alto.
2. Uma escrita ocorre durante a superposição de um $\overline{CS}$ baixo a um $\overline{WE}$ baixo. t_W é medido a partir da última transição negativa de $\overline{CS}$ ou $\overline{WE}$ até a primeira transição positiva de $\overline{CS}$ ou $\overline{WE}$.

FORMAS DE ONDA

CICLO DE LEITURA[3]

CICLO DE ESCRITA

NOTAS:

3. $\overline{WE}$ está alto para um ciclo de leitura.
4. Se a transição negativa de $\overline{CS}$ ocorrer simultaneamente com a transição negativa de $\overline{WE}$, os buffers de saída permanecerão em estado de alta impedância.
5. $\overline{WE}$ deve estar alto durante todas as transições de endereços.

APÊNDICE C5

8212

8 — Bit Input/Output port (Porta de E/S)

- **Registro de dados e buffer de 8 bits paralelos**
- **Flip-flop para pedido de tratamento para geração de interrupção**
- **Baixa corrente de carga de entrada – 0,25 mA máx**
- **Saídas Tri-State**
- **Saídas drenam 15 mA**
- **V_{OH} de 3,65V para interfacear direto com CPU's 8008, 8080A ou 8085A.**
- **Limpeza de registro assíncrona**
- **Substitui buffers, latches e multiplexadores em sistemas de computação**
- **Reduz o tamanho do sistema**

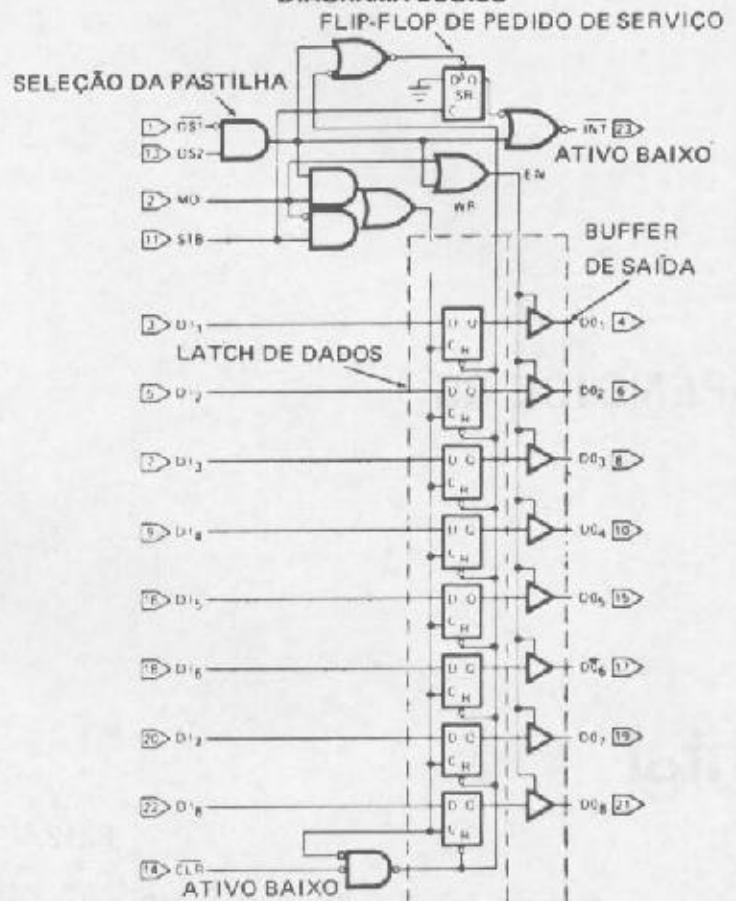
A 8212 (porta de E/S) consiste de um latch (N.T. circuito armazenador temporário de dados) de 8 bits com buffer de saída Tri-State com lógica de seleção e controle. Um flip-flop para pedido de tratamento é incluído para geração e controle de interrupções para o microprocessador. O componente é por natureza multiutilizável. Pode ser utilizado para implementar latches, buffers controláveis ou até multiplexadores. Inclusive, todas as principais funções de periféricos e de E/S de um sistema de microcomputador podem ser implementadas com este componente.

DESCRIÇÃO FUNCIONAL

Latch de Dados (Data Latch)

Os 8 flip-flops que compõem o latch de dados são do tipo "D". A saída "Q" do flip-flop será idêntica à entrada "D", enquanto a entrada clock (C) estiver alta. O armazenamento ocorre quando o clock (C) volta a ser baixo.

O dado armazenado é limpo por um "reset" assíncrono ($\overline{CLR}$). (Nota: o clock (C) superpõe-se ao reset ($\overline{CLR}$).

DI_1-DI_8	DATA IN
DO_1-DO_8	DATA OUT
$\overline{DS_1}$, DS_2	DEVICE SELECT
MD	MODE
STB	STROBE
$\overline{INT}$	INTERRUPT (ACTIVE LOW)
$\overline{CLR}$	CLEAR (ACTIVE LOW)

Buffer de Saída

As saídas do latch de dados (Q) são ligadas a buffers de saída TRI-STATE não inversores. Estes buffers possuem uma linha de controle comum (EN). Este controle tanto habilita o buffer para transmitir os dados das saídas do latch de dados (Q), como desabilita o buffer, forçando as saídas $Do_1 - Do_8$ a um estado de alta impedância (TRI-STATE).

O estado de alta impedância permite ao projetista conectar o 8212 diretamente nas barras bidirecionais do sistema.

Lógica de Controle

O 8212 tem entradas de controle DS1, DS2, MD e STB. Estas entradas são utilizadas para controlar a seleção da pastilha, o armazenamento de dados, o estado do buffer de saída e o flip-flop (SR) de pedido de tratamento (interrupções).

$\overline{DS1}$, $\overline{DS2}$ (Device Select – Seleção da Pastilha)

Estas duas entradas são utilizadas para a seleção da pastilha. Quando $\overline{DS1}$ está baixo e $\overline{DS2}$ alto, a pastilha é selecionada ($\overline{DS1} \cdot \overline{DS2}$). Neste estado, o buffer de saída é habilitado e o flip-flop (SR) é levado a estado "1" assincronamente.

MD (Modo)

Esta entrada é utilizada para controlar o estado do buffer de saída e para determinar a fonte da entrada de clock (C) do latch de dados.

Quando MD está alto (modo de saída), os buffers da saída são habilitados e o clock (C) do data latch é fornecido pela lógica de seleção ($\overline{DS1} \cdot \overline{DS2}$).

Quando MD está baixo (modo de entrada), o estado do buffer de saída é determinado pela lógica de seleção ($\overline{DS1} \cdot \overline{DS2}$) e o clock (C) do latch de dados é fornecido pela entrada STB (Strobe).

STB (Strobe)

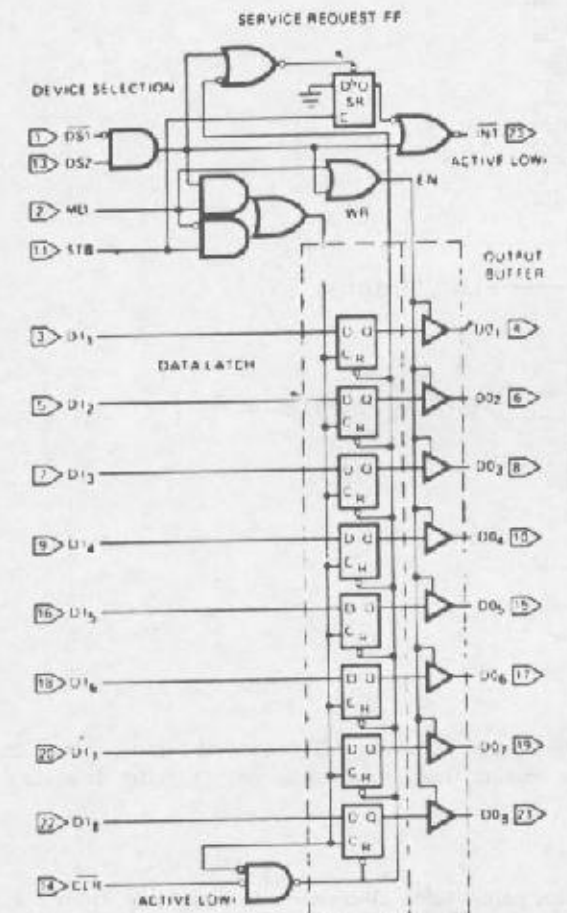
Esta entrada é utilizada como clock (C) para o latch de dados no modo de entrada (MD = 0) e para levar o flip-flop (SR) ao estado "0" simultaneamente.

Observe que o flip-flop (SR) é disparado por transição negativa.

Flip-flop de pedido de tratamento (SR)

O flip-flop (SR) é utilizado para gerar e controlar pedidos de interrupção nos sistemas de microcomputadores. Ele é levado a "1" assincronamente pelo pulso de $\overline{CLR}$ ativo baixo. Ao estar em estado "1", o flip-flop está em estado de não-interrupção.

A saída do flip-flop SR (Q) está conectada à entrada inversora de uma porta NOR. A entrada desta porta é não inversora e está conectada à lógica de seleção da pastilha ($\overline{DS1} \cdot DS2$).

A saída da porta NOR ($\overline{INT}$) é ativo baixo (estado de interrupção) para conectar em entrada ativo baixo de circuitos de geração de prioridade (N.T. controladores de interrupção, por exemplo).

STB	MD	($\overline{DS_1} \cdot DS_2$)	DATA OUT EQUALS
0	0	0	3 STATE
1	0	0	3 STATE
0	1	0	DATA LATCH
1	1	0	DATA LATCH
0	0	1	DATA LATCH
1	0	1	DATA IN
0	1	1	DATA IN
1	1	1	DATA IN

$\overline{CLR}$ – RESETS DATA LATCH
SETS SR FLIP FLOP
(NO EFFECT ON OUTPUT BUFFERS)

$\overline{CLR}$	($\overline{DS_1} \cdot DS_2$)	STB	*SR	$\overline{INT}$
0	0	0	1	1
0	1	0	1	0
1	1	↓	0	0
1	1	0	1	0
1	0	0	1	1
1	1	↑	1	0

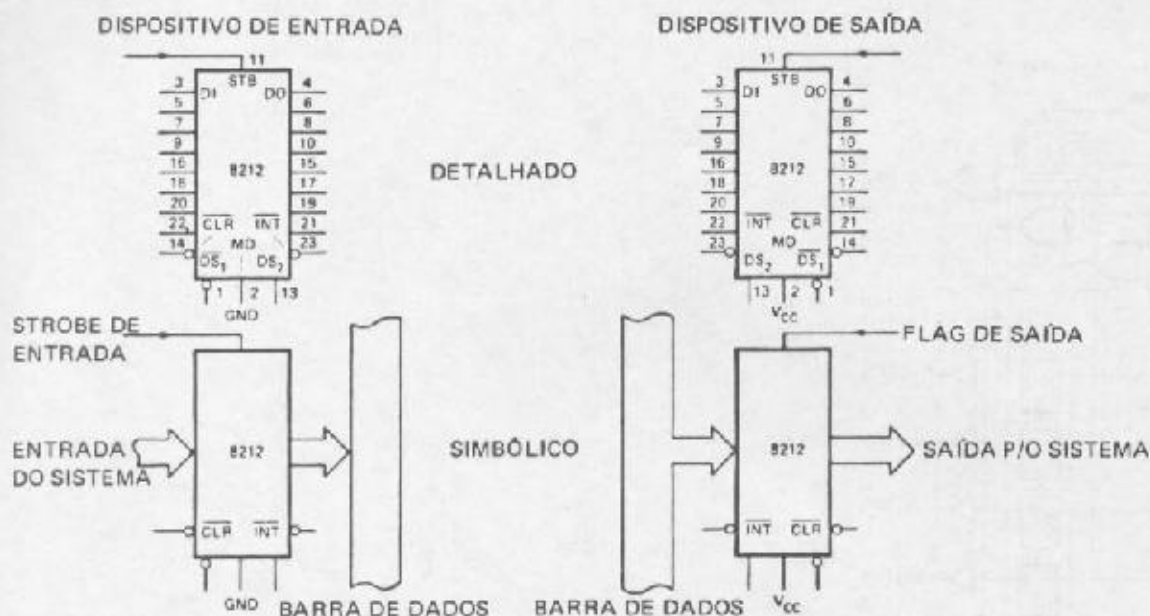
*INTERNAL SR FLIP FLOP

Aplicações do 8212 em sistema de microcomputadores

I – Símbolo esquemático básico
II – Porta buferizada*
III – Barra bidirecional de dados buferizada*
IV – Porta de entrada com interrupção
V – Porta de instrução de interrupção
VI – Porta de saída
VII – Latch de *status* do 8080A
VIII – Latch de endereços do 8085A

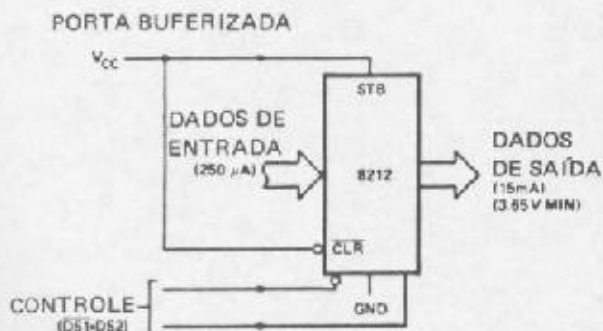
* (N.T. – Buferizada – neologismo proveniente da palavra *Buffer*, que significa que um circuito teve sua capacidade de corrente aumentada).

I. Símbolo esquemático básico

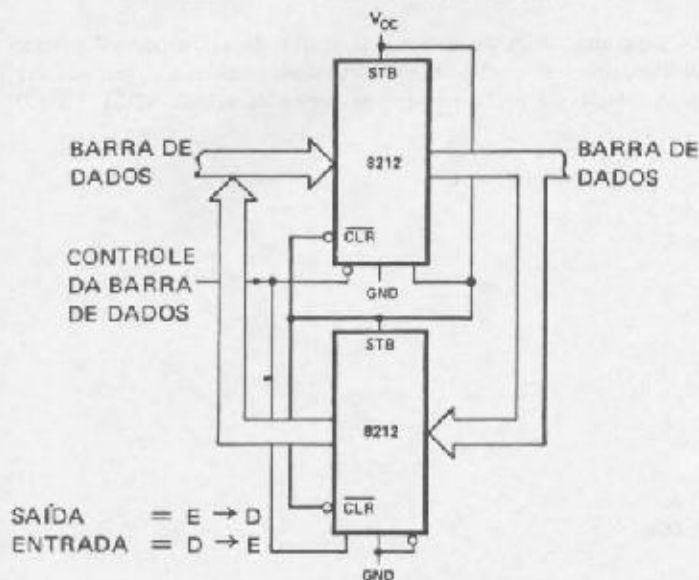
Dois exemplos para se desenhar o 8212 em esquemáticos: (1) Acima uma vista detalhada mostrando a pinagem e (2) abaixo uma vista simbólica mostrando a entrada ou saída do sistema como uma barra de sistema (contendo 8 linhas paralelas). A saída para a barra de dados é simbólica para referenciar 8 linhas paralelas.

II. Porta buferizada (Tri-State)

A utilização mais simples do 8212 é como uma porta buferizada. Amarrando-se o sinal de modo (MD) em nível baixo e o sinal de strobe (STB) em nível alto, o latch de dados funciona como uma porta lógica comum (não armazena dados). O buffer de saída é habilitado então pela lógica de seleção da pastilha ($\overline{DS1} \cdot \overline{DS2}$).

Quando a pastilha não está selecionada, as saídas ficam em tri-state.

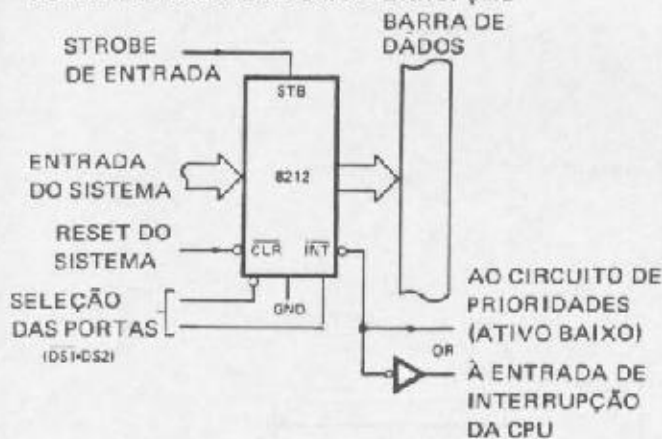
Quando a pastilha é selecionada, os dados de entrada são diretamente transferidos para a saída. A corrente de entrada de dados é de 250 μA. As saídas de dados podem drenar 15 mA. A tensão de saída mínima é de 3,65 volts.

III. Barra de dados bidirecional buferizada

Um par de 8212 com as entradas ligadas nas saídas podem ser utilizadas como um módulo de barra de dados bidirecional. Os componentes são controlados pelo controle de entrada da barra de dados que é conectado a $\overline{DS1}$ no primeiro 8212 e a $\overline{DS2}$ no segundo. Enquanto um campo fica ativo e funcionando como uma porta buferizada, o outro fica em tri-state. Este é um circuito bem útil para projetos de pequeno porte.

BARRA DE DADOS BIDIRECIONAL BUFERIZADA

IV. Porta de entrada com interrupção

Esta utilização do 8212 é como uma porta de entrada para o sistema que recebe um sinal externo de Strobe para avisar que foram armazenados dados dentro dela que, por sua vez, limpa o flip-flop SR, gerando um pedido de interrupção ao processador. O processador é, então, desviado para uma rotina de tratamento, identifica a porta, seleciona a pastilha e habilita a entrada de dados para a barra do sistema.

PORTA DE ENTRADA COM INTERRUPÇÃO

STROBE DE ENTRADA
BARRA DE DADOS
STB
ENTRADA DO SISTEMA
8212
RESET DO SISTEMA
CLR
INT
SELEÇÃO DAS PORTAS (DS1·DS2)
GND
AO CIRCUITO DE PRIORIDADES (ATIVO BAIXO)
OR
À ENTRADA DE INTERRUPÇÃO DA CPU

V. Porta de instrução de interrupção

A 8212 pode ser usada para gerar uma instrução de interrupção (normalmente instruções de Restart) e colocá-la na barra de dados. A pastilha é habilitada pelo sinal de reconhecimento de uma interrupção e pelo sinal de seleção da porta. Este sinal é normalmente ligado à terra. (DS1 pode ser utilizado para multiplexar instruções da interrupção de várias portas para uma barra comum.)

VI. Porta de saída (com protocolo)

A 8212 pode ser utilizada para transmitir dados da barra de dados para uma saída do sistema. O strobe de saída pode ser usado como sinal de protocolo tal como "recebimento de dado", vindo do dispositivo ao qual o sistema está dando saída. Ele, por sua vez, pode interromper o sistema significando que o dado foi recebido. A seleção da porta provém da lógica de seleção ($\overline{DS1}$ * $\overline{DS2}$).

VII. Latch de *status* do 8080A (armazenador de estado)

Aqui, o 8212 é utilizado como latch de estado para um sistema de microcomputador com 8080A. As entradas do 8212 são conectadas diretamente à barra de dados do 8080A. O diagrama de tempo mostra que quando ocorrem o sinal Sync (que está conectado à DS2) e o sinal de fase 1 (01) (sinal proveniente do gerador de clock) então o estado será armazenado no 8212.

Nota: O sinal de modo é amarrado em "1", de maneira que a saída do latch fica o tempo todo habilitada.

É mostrado que as duas áreas de interação são a barra bidirecional de dados do microprocessador e a barra de controle.

VIII. Latch de endereços de baixa ordem do 8085A

O microprocessador 8085A utiliza uma barra de endereços multiplexados que contém os 8 bits menos significativos (baixa ordem) do endereço, durante a primeira parte do ciclo de máquina. A mesma barra contém dados numa etapa posterior do mesmo ciclo. Um sinal habilitador do latch de endereços (ALE) é gerado pelo 8085A a fim de ser usado pelo 8212, para armazenar o endereço para que este possa estar disponível durante todo o ciclo de máquina.

Nota: Nesta configuração o sinal de modo (MD) é amarrado alto (V_{CC}), mantendo o buffer de saída o tempo todo habilitado.

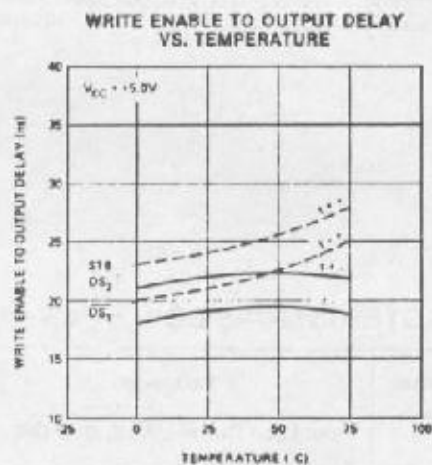
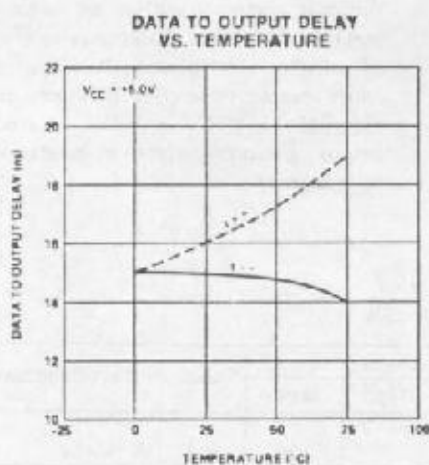
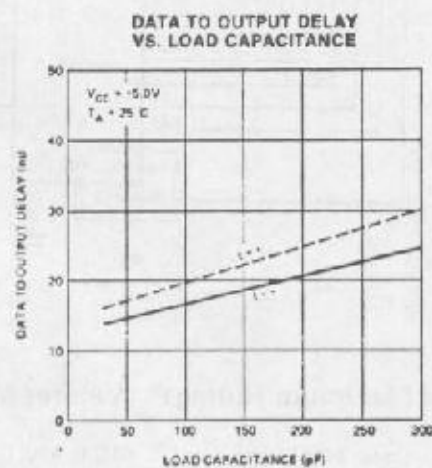
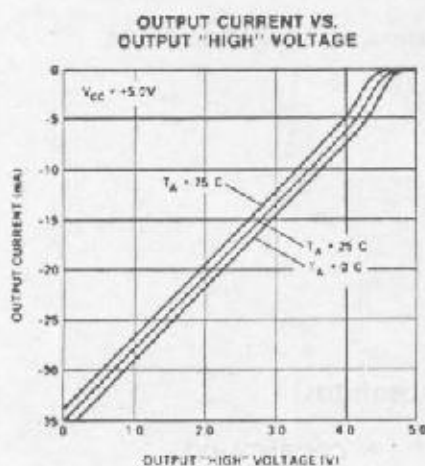
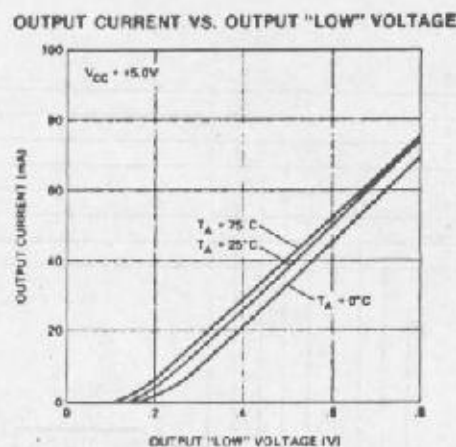
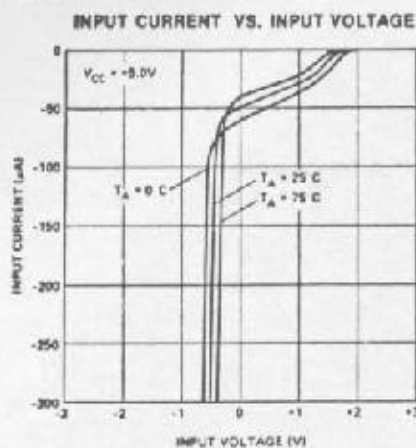
Absolute Maximum Ratings* (Valores Máximos Absolutos)

Temperature Under Bias Plastic .... 0° C to +70° C
Storage Temperature .... -65° C to +160° C
All Output or Supply Voltages .... -0.5 to +7 Volts
All Input Voltages .... -1.0 to 5.5 Volts
Output Currents 100mA

* COMENTÁRIO
Esforços maiores do que os especificados nos "Valores Máximos Absolutos" (Absolute Max. Ratings) podem danificar permanentemente o componente. Estes são apenas valores máximos de esforço, e a operação funcional do componente nestas condições não está prevista. A exposição aos valores máximos absolutos por longos períodos afeta a confiabilidade do componente.

CARACTERÍSTICAS CC $T_A = 0°C$ a $+75°C$, $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Limits Min.	Limits Typ.	Limits Max.	Unit	Test Conditions
I_F	Input Load Current, ACK, DS2, CR, DI1-DI8 Inputs			-.25	mA	$V_F = .45V$
I_F	Input Load Current MD Input			-.75	mA	$V_F = .45V$
I_F	Input Load Current DS1 Input			-1.0	mA	$V_F = .45V$
I_R	Input Leakage Current, ACK, DS, CR, DI1-DI8 Inputs			10	μA	$V_R \le V_{CC}$
I_R	Input Leakage Current MO Input			30	μA	$V_R \le V_{CC}$
I_R	Input Leakage Current DS1 Input			40	μA	$V_R \le V_{CC}$
V_C	Input Forward Voltage Clamp			-1	V	$I_C = -5mA$
V_{IL}	Input "Low" Voltage			.85	V	
V_{IH}	Input "High" Voltage	2.0			V	
V_{OL}	Output "Low" Voltage			.45	V	$I_{OL} = 15mA$
V_{OH}	Output "High" Voltage	3.65	4.0		V	$I_{OH} = -1mA$
I_{SC}	Short Circuit Output Current	-15		-75	mA	$V_O = 0V$, $V_{CC} = 5V$
$\|I_O\|$	Output Leakage Current High Impedance State			20	μA	$V_O = .45V/5.25V$
I_{CC}	Power Supply Current		90	130	mA	

CARACTERÍSTICAS TÍPICAS

CARACTERÍSTICAS CA $T_A = 0°C$ a $+70°C$, $V_{CC} = +5V \pm 5\%$

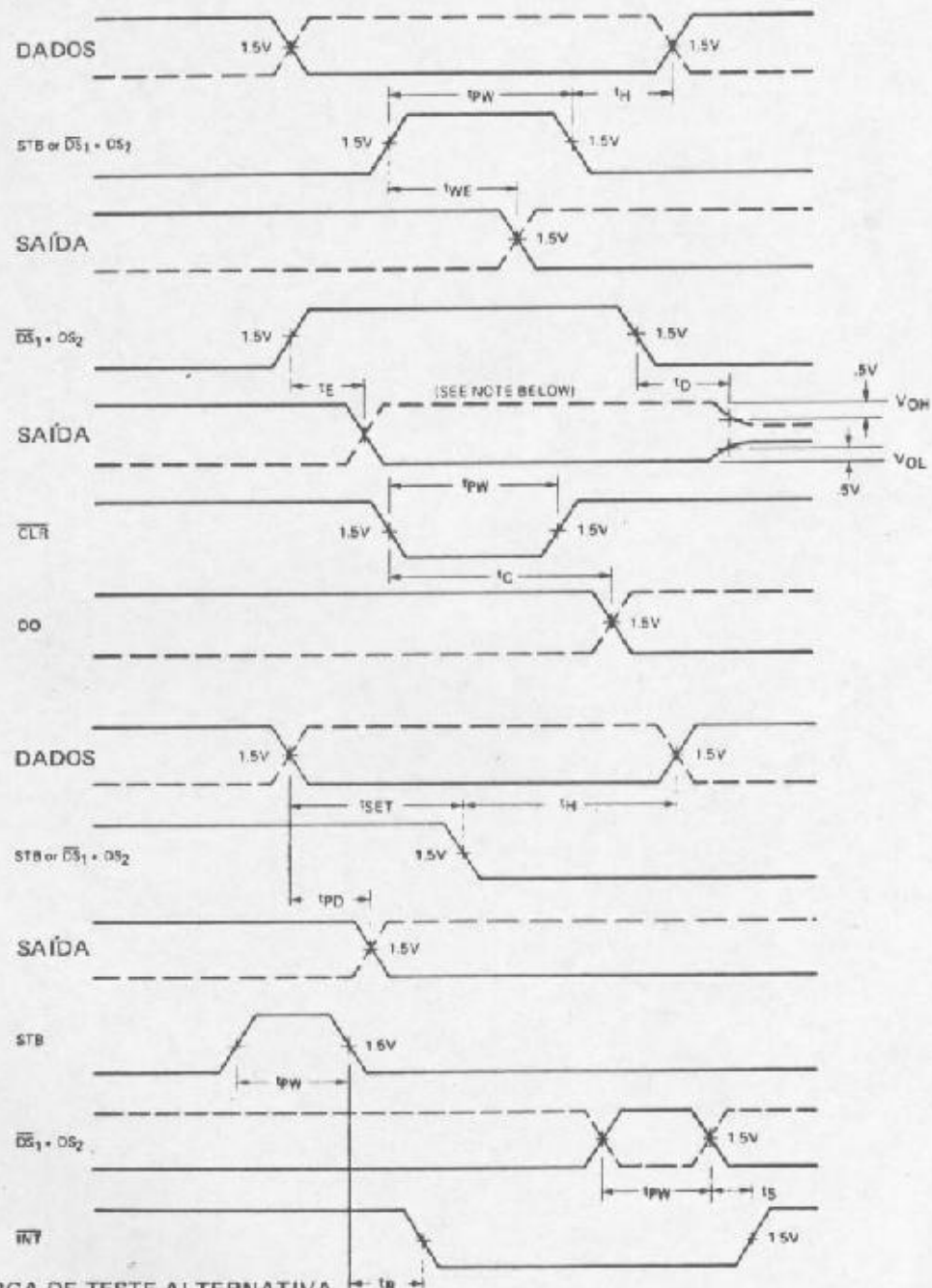
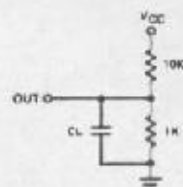
Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
t_{PW}	Pulse Width	30			ns	
t_{PD}	Data to Output Delay			30	ns	Note 1
t_{WE}	Write Enable to Output Delay			40	ns	Note 1
t_{SET}	Data Set Up Time	15			ns	
t_H	Data Hold Time	20			ns	
t_R	Reset to Output Delay			40	ns	Note 1
t_S	Set to Output Delay			30	ns	Note 1
t_E	Output Enable/Disable Time			45	ns	Note 1
t_C	Clear to Output Delay			55	ns	Note 1

CAPACITÂNCIA* F = 1MHz, V_{BIAS} = 2.5V, V_{CC} = +5V, T_A = 25°C

Symbol	Test	Limits	
		Typ.	Max.
C_{IN}	DS_1 MD Input Capacitance	9pF	12pF
C_{IN}	DS_2, CK, ACK, DI_1-DI_8 Input Capacitance	5pF	9pF
C_{OUT}	DO_1-DO_8 Output Capacitance	8pF	12pF

Este parâmetro é amostrado e não testado a 100%.

8212

DIAGRAMA DE TEMPO

NOTA: CARGA DE TESTE ALTERNATIVA

CARACTERÍSTICAS DE CHAVEAMENTO

Condições de Teste

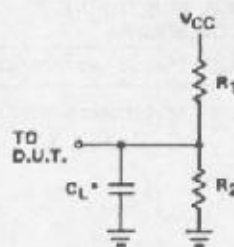
Input Pulse Amplitude = 2.5V
Input Rise and Fall Times 5ns
Between 1V and 2V Measurements made at 1.5V
with 15mA and 30pF Test Load

Note 1:

Test	C_L*	R_1	R_2
t_{PD}, t_{WE}, t_R, t_S, t_C	30pF	300Ω	600Ω
t_E, ENABLE↑	30pF	10KΩ	1KΩ
t_E, ENABLE↓	30pF	300Ω	600Ω
t_E, DISABLE↑	5pF	300Ω	600Ω
t_E, DISABLE↓	5pF	10KΩ	1KΩ

*Includes probe and jig capacitance.

Test Load

15mA & 30pF

*INCLUDING JIG & PROBE CAPACITANCE

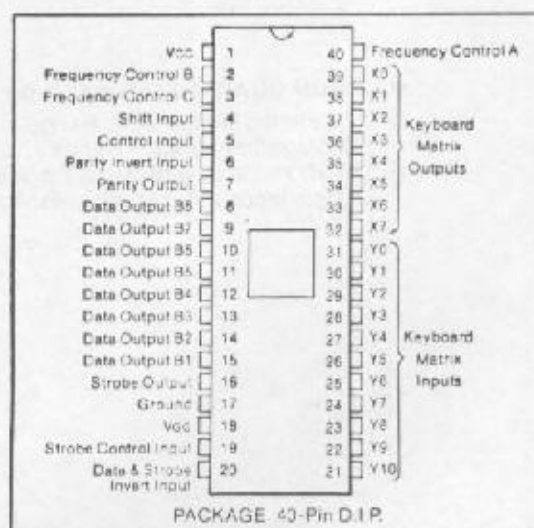
APÊNDICE C6

KR2376-XX

Keyboard Encoder Read Only Memory (ROM Codificadora para Teclado)

FACILIDADES

- Saídas diretamente compatíveis com TTL/DTL ou redes lógicas MOS
- Controle externo para selecionar polaridade da saída
- Controle externo para selecionar paridade par ou ímpar
- Operação com duas teclas seqüenciais
- Tecla maiúscula/minúscula
- Códigos programáveis com troca simples de máscara
- Oscilador interno
- Circuito de retardo controlado externamente para eliminar o efeito de Bounce
- Apenas um integrado é necessário para montar um teclado inteiro
- Proteção contra carga estática em todos os pinos de entrada e saída
- Proteção total para o circuito com camada de vidro

CONFIGURAÇÃO DOS DADOS

DESCRIÇÃO GERAL

O SMC KR2376-XX é uma ROM de 2376 bits com toda a lógica adicional necessária para transformar os sinais provenientes de um teclado de pólo simples num código utilizável de 9 bits. As saídas de dados e de Strobe são diretamente compatíveis com lógicas TTL/DTL ou MOS, sem utilização de nenhum componente para interfaceamento.

O KR2376-XX é fabricado com tecnologia canal-P de baixo limiar e contém 2942 transistores num único componente monolítico, encontrado em encapsulamento de 40 pinos Dual-In-Line.

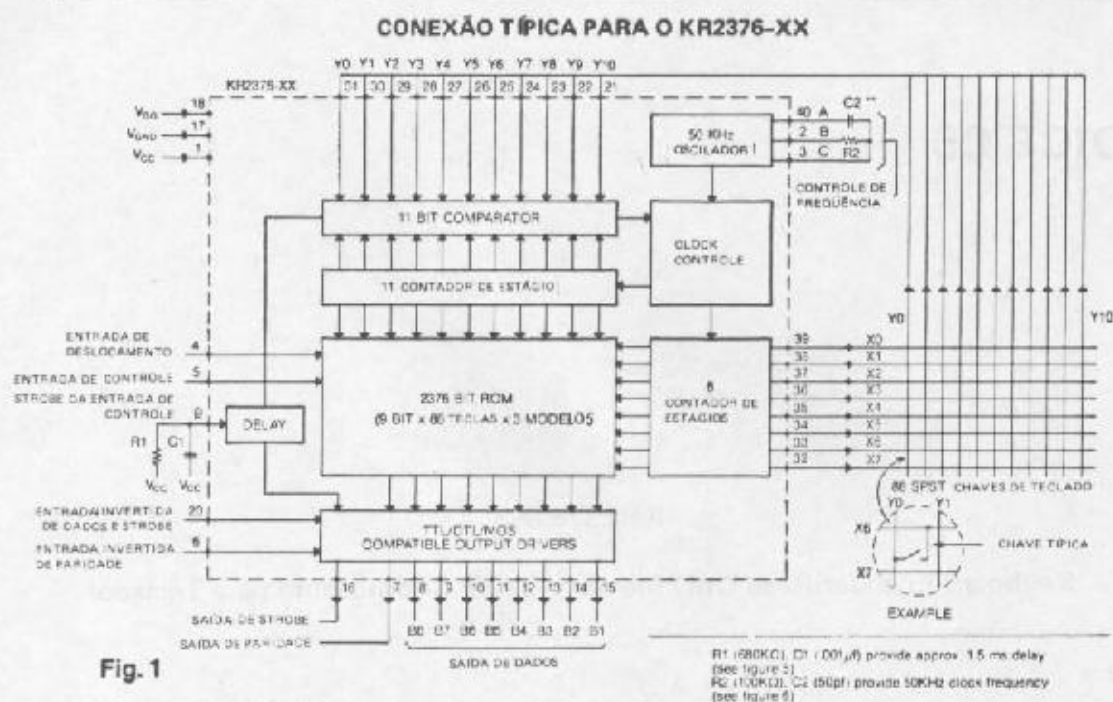



Fig. 1

R1 (680KΩ), C1 (.001μf) provide approx. 1.5 ms delay (see figure 5)
R2 (100KΩ), C2 (50pf) provide 50KHz clock frequency (see figure 6)

MAXIMUM GUARANTEED RATINGS (Valores Máximos Garantidos)

Operating Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C
GND and V_{GG}, with respect to V_{CC}	−20V to +0.3V
Logic Input Voltages, with respect to V_{CC}	−20V to +0.3V

* COMENTÁRIO
Esforços maiores do que os especificados podem danificar permanentemente o componente. Estes são apenas valores máximos de esforço e a operação funcional do componente nestas condições não está prevista.

DESCRIÇÃO DE OPERAÇÃO

O KR2376-XX contém (veja figura 1) uma ROM de 2376 bits, contadores de 8 e de 11 estágios, um comparador de 11 bits, um oscilador, um circuito de retardo controlado externamente para eliminar efeito de Bounce, e saídas compatíveis com TTL/DTL/MOS.

A parte de ROM da pastilha é uma memória de 264 por 9 bits agrupadas em três conjuntos de 88 palavras de 9 bits. Níveis apropriados nas entradas de controle e deslocamento selecionam um dos três conjuntos de 88 palavras. Os dois contadores servem para endereçar cada uma das 88 palavras de cada grupo. Sendó assim, o endereço da ROM é formado combinando-se as entradas de controle e deslocamento (Shift) com os dois contadores.

As saídas externas do contador de 8 estágios e as entradas externas do comparador de 11 bits são fixadas no teclado, de maneira a formar uma matriz X-Y com 88 chaves do teclado (teclas) nos cruzamentos dos fios. Na condição de espera, quando nenhuma tecla é apertada, os dois contadores são incrementados e endereçam a ROM seqüencialmente. A ausência de um pulso de Strobe na saída indica que o conteúdo das saídas de dados não é válido neste instante.

CARACTERÍSTICAS ELÉTRICAS

($T_A = 0°C$ a $+70°C$, $V_{CC} = +5V \pm 0.5V$, $V_{GG} = -12V \pm 1.0V$, a menos que especificado em contrário)

Características	Mín.	Tipo	Máx.	Unid.	Conditions
CLOCK	20	50	100	KHz	see fig.1 footnote (**) for typical R-C values
DATA INPUT					
Logic "0" Level			+0.8	V	
Logic "1" Level	Vcc-1.5			V	
Input Capacitance			10	pf	
INPUT CURRENT					
*Control, Shift & Y0 thru Y10	10	100	140	μA	$V_{IN} = +5.0V$
*Control, Shift & Y0 thru Y10	5	30	50	μA	V_{IN} = Ground
Data Invert, Parity Invert		.01	1	μA	$V_{IN} = -5.0V$ to $+5.0V$
DATA OUTPUT & X OUTPUT					
Logic "0" Level			+0.4	V	$I_{OL} = 1.6mA$ (see fig. 7)
Logic "1" Level	Vcc-1.0			V	$I_{OH} = 100\ \mu A$
POWER CONSUMPTION		140	200	mW	Nom. Power Supp. Voltages (see fig. 8)
SWITCH CHARACTERISTICS					
Minimum Switch Closure	see timing diagram-fig. 2				
Contact Closure Resistance between X1 and Y1			300	Ohm	
Contact Open Resistance between X1 and Y1	1×10^7			Ohm	

* Entrada como resistor interno para V_{GG}

DIAGRAMA DE TEMPO

Fig. 2

CONEXÕES DA FONTE PARA OPERAÇÃO TTL/DTL

CONEXÕES DA FONTE PARA OPERAÇÃO MOS

Fig. 3

DRIVER DE SAÍDA DO TECLADO

ESTÁGIO DE ENTRADA DO TECLADO

Fig. 4

Quando uma tecla é apertada, um único caminho é fechado entre uma das saídas do contador de 8 estágios ($X_0 - X_7$) e uma das entradas do comparador de 11 bits ($Y_0 - X_{10}$). Após um número definido de ciclos de clock, ocorrerá uma condição onde um nível do caminho fechado até o comparador será igual a um nível na entrada correspondente deste comparador vindo do contador de 11 estágios. Quando isto ocorre, o comparador gera um sinal para o controle de clock e para a saída de Strobe (passando pelo circuito de retardo). O controle de clock pára de incrementar os contadores, e as saídas de dados ($B_1 - B_9$) se estabilizam com o código de 9 bits selecionado, indicados como válidos pelo pulso de Strobe. As saídas de dados se mantêm estáveis mesmo se a tecla foi liberada.

Como uma facilidade adicional, existem duas entradas para controle externo de polaridade das saídas de dados. "Parity Invert" (pino 6) permite o controle de "Parity Output" (pino 7) enquanto o "Data e Strobe Invert Input" (pino 20) permite o controle da polaridade das saídas de dados $B_1 - B_8$ (pinos de 8 a 15) e do "Strobe Output" (pino 16).

PADRÕES ESPECIAIS

Como o código gerado por tecla é definido durante a fabricação do componente, a codificação pode ser alterada para servir a aplicações especiais do teclado. Até 264 códigos de 8 bits (mais um bit de paridade) podem ser programados na ROM do KR2376-XX, cobrindo assim a maioria dos códigos utilizados comumente, tais como ASCII, EBCDIC, SELECTRIC etc. assim como diversos códigos especiais. O código ASCII é fornecido como padrão standard. Para padrões especiais, veja a figura 9.

Fig. 5 Fig. 6 Fig. 7 Fig. 8

MAPA DE CONFIGURAÇÃO DE CÓDIGO
KR2376-ST
8 Bit ASCII, odd parity

DADO INVERTIDO TABELA VERDADE

DATA & STROBE INVERT INPUT (Pin 20)	CODE ASSIGNMENT CHART	DATA OUTPUTS (B1-B8)
1	1	0
0	1	1
1	0	1
0	0	0

STROBE INVERTIDO TABELA VERDADE

DATA & STROBE INVERT INPUT (Pin 20)	INTERNAL STROBE	STROBE OUTPUT (Pin 16)
1	1	0
0	0	0
1	0	1
0	1	1

PARIDADE DA TABELA VERDABE

PARITY INVERT INPUT (Pin 6)	CODE ASSIGNMENT CHART	PARITY OUTPUT (Pin 7)
1	1	0
0	1	1
1	0	1
0	0	0

MODO DE SELEÇÃO

$\bar{S}$ $\bar{C}$ = N
S $\bar{C}$ = S
$\bar{S}$ C = C
S C = INVALID (SPURIOUS DATA)

Fig. 9

N S C — B1 B2 B3 B4 B5 B6 B7 B8 (X0) — PARITY BIT
1 2 3 4 5 6 7 8 9
Y9 → NORMAL / SHIFT / CONTROL

(Code representative of key depression at location X0-Y9 and proper mode selection)

N = Normal Mode
S = Shift Mode
C = Control Mode
■ = Output Logic "1" (see data B1-B8)
Logic "1" = +5.0V
Logic "0" = Ground

APÊNDICE C7

COM 2502
COM 2017
COM 2502/H
COM 2017/H

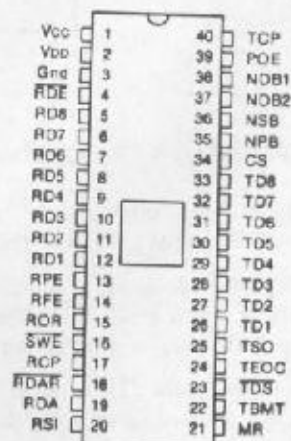
Universal Asynchronous Receiver/Transmiter

UART

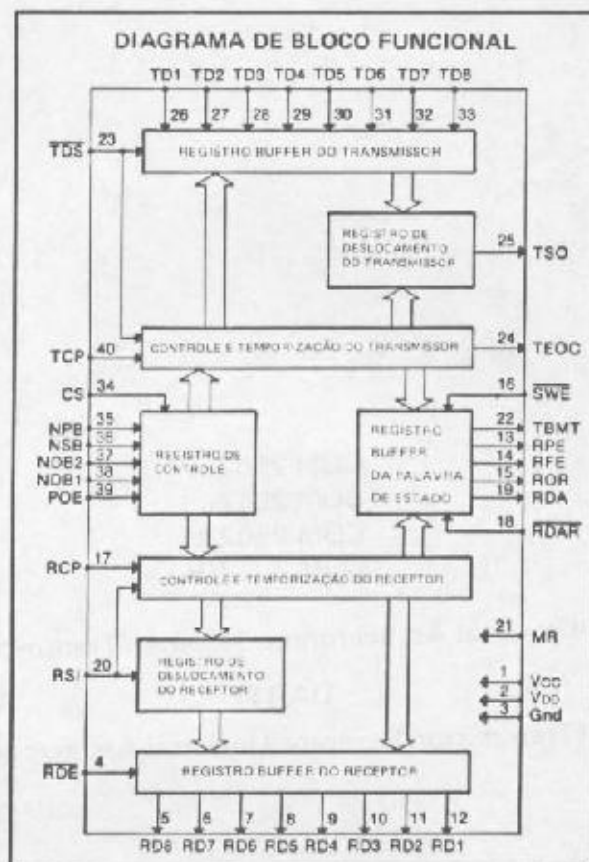
(Transmissor/Receptor Universal Assíncrono)

FACILIDADES

CONFIGURAÇÃO DOS PINOS

- Diretamente compatível com TTL – não necessita de circuitos de interfaceamento
- Operação em Full-Duplex ou Half-Duplex – pode transmitir e receber simultaneamente com taxas de Baud diferentes
- Dupla buferização – elimina necessidade de temporização externa
- Verificação do bit de partida diminui taxa de erros
- Totalmente programável – tamanho da palavra de dados, modo de paridade, número de bits de parada; um, um e meio, ou dois
- Master Reset – limpa todas as saídas de estado
- Saídas Tri-State – destinadas à estrutura de barras
- Baixo consumo – necessidades mínimas de potência
- **Operação em alta velocidade – 40K bauds, 200 ns de strobe**
- **Entradas protegidas – eliminam problemas de manuseio**
- **Encapsulamento cerâmico ou plástico – fácil inserção em cartões**

DESCRIÇÃO GERAL

O UART é um circuito MOS/LSI, que desempenha todas as funções de transmissão e recepção associadas à comunicação de dados assíncronos. Este circuito é fabricado usando tecnologia de baixo consumo CANAL-P óxido-nitrido da SMC. O modo duplex, a taxa de Baud, o tamanho da palavra, o modo de paridade e o número de bits de parada são programáveis independentemente utilizando-se os controles externos. Podem existir 5, 6, 7 ou 8 bits de dados, paridade par/ímpar ou sem paridade, 1 ou 2 bits de parada ou 1,5 bits de parada, quando se utiliza um código de 5 bits da COM 2017 ou COM 2017/II.

O UART pode operar tanto em Full-Duplex quanto em Half-Duplex. Estas facilidades de programação possibilitam ao usuário interfacear com todos os periféricos assíncronos.

DESCRIÇÃO DE OPERAÇÃO – TRANSMISSOR

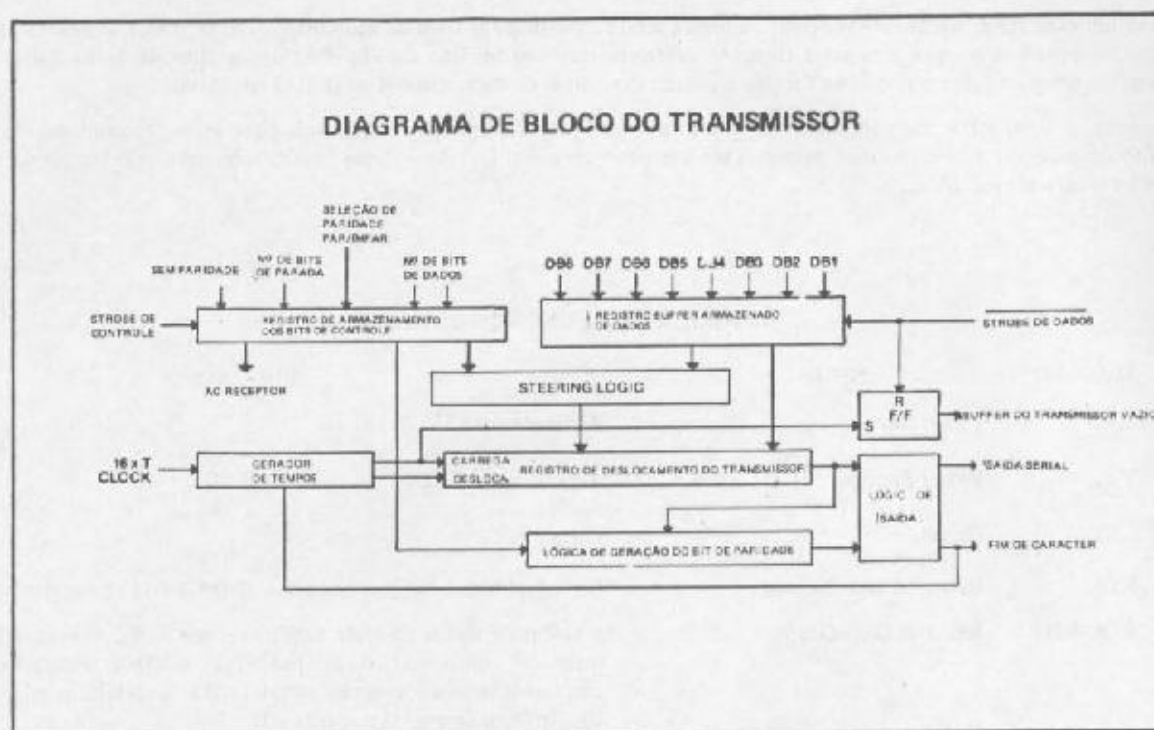
De início, a alimentação é ligada, um relógio cuja freqüência é 16 vezes maior do que a taxa de baud desejada é ligado e o Master Reset é pulsado. Nestas condições, TBMT, TEOC e TSO ficam em estado alto ("1").

Quando TBMT e o TEOC estão altos, os bits de controle podem ser configurados. Feito isto, os bits de dados podem ser configurados. Normalmente, os bits de controle são configurados antes dos de dados. Entretanto, se as especificações de tamanho de pulso não forem violadas, $\overline{TSD}$ e CS poderão ocorrer simultaneamente.

Uma vez que o strobe de dados ($\overline{TDS}$) tenha sido pulsado, o sinal TBMT vai para estado baixo, indicando que o registro dos bits de dados já está cheio e que não pode receber novos dados.

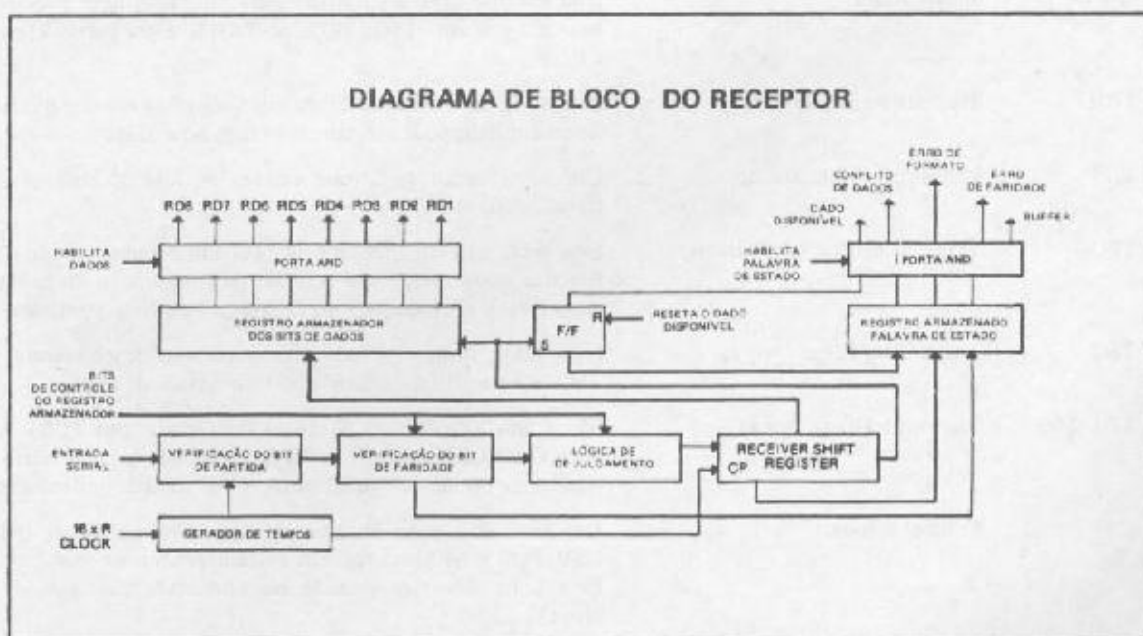
Caso o registro de deslocamento do transmissor esteja transmitindo algum dado carregado anteriormente, o sinal TBMT permanece baixo. Se o registro de deslocamento do transmissor estiver vazio, ou quando ele acaba de transmitir o dado anterior, o dado do buffer será carregado imediatamente no registro de deslocamento do transmissor e a transmissão de dados terá início. TSO vai para baixo (o bit de partida), TEOC vai para baixo, TBMT vai para o alto, indicando que o dado do buffer foi carregado no registro de deslocamento e que o registro de dados do buffer já pode receber um novo dado.

Caso um novo dado seja carregado no registro do buffer de dados neste instante, TBMT vai para baixo e fica neste estado até que a transmissão atual se complete. Pode-se esperar o tempo de transmissão de um caracter inteiro antes de carregar o próximo caracter, sem que haja perda na velocidade de transmissão. Esta é uma vantagem da dupla buferização. (N.T. a palavra buffer aqui é utilizada significando um registro de armazenamento.)

A transmissão de dados se processa de um modo ordenado: bit de partida, bits de dados, bit de paridade (caso selecionado) e o(os) bits de parada. Quando o último bit de parada ficar na linha durante o tempo de um bit, TEOC vai para alto. Se TBMT estiver alto, o transmissor estará inoperante e, caso se deseje, novos bits de controle podem ser carregados antes da próxima transmissão de dados.

DESCRIÇÃO DE OPERAÇÃO – RECEPTOR

De início, a alimentação é ligada, um relógio cuja freqüência é 16 vezes a taxa de baud desejada é ligado e o Master-Reset é pulsado. O sinal de dado disponível (RDA) está baixo. Os bits de controle para o transmissor e o receptor são os mesmos.

A recepção de dados começa quando a linha de entrada serial transiciona de "1" para "0". Caso a linha RS1 permaneça em "0" pelo tempo de mais 1/2 bit, então é feita a verificação de um bit de partida genuíno. Caso a linha volte a "1" antes do tempo de 1/2 bit, o processo para verificação do bit de partida reinicia. A transição de "1" para "0" deve ocorrer para possibilitar a verificação do bit de partida. Uma vez feita esta verificação, a recepção de dados procede de uma maneira ordenada: bit de partida verificado e recebido, bits de dados recebidos, bit de paridade recebido (se houver) e os bits de parada recebidos.

Caso o bit de paridade transmitido não seja igual ao recebido, o flip-flop de erro de paridade do buffer da palavra de estado é colocado alto, indicando erro de paridade. Entretanto, se o modo de paridade não for selecionado, este flip-flop será mantido incondicionalmente baixo, inibindo a indicação de paridade. Caso um bit de parada não seja recebido devido a um erro de formação do caracter, o flip-flop de erro de formato é colocado alto para indicar um erro de formato (estrutura).

Uma vez que um caracter é totalmente recebido, a lógica interna verificará o sinal de dado disponível (RDA). Caso neste instante RDA esteja alto, o receptor assume que o caracter recebido anteriormente não foi lido e o flip-flop de conflito de dados é colocado alto. A única maneira do receptor saber que o dado foi lido é colocando o sinal de dado disponível (RDA) em baixo.

Neste momento, o sinal RDA vai para alto, indicando que todas as saídas estão disponíveis para serem examinadas. O registro de deslocamento do receptor está disponível para receber um novo caracter. Devido à dupla buferização, tem-se o tempo de um caracter inteiro para ler o caracter recebido.

DESCRIÇÃO DA FUNÇÃO DOS PINOS

Nº PINO	SÍMBOLO	NOME	FUNÇÃO
1	V_{CC}	Power Supply	Alimentação +5V
2	V_{DD}	Power Supply	Alimentação – 12V
3	GND	Ground	Terra
4	$\overline{RDE}$	Received Data Enable	Um nível baixo habilita as saídas (RD8-RD11) do registro de recepção
5-12	RD8-RD1	Receiver Data Outputs	8 saídas de dados tri-state habilitadas por $\overline{RDE}$. As saídas de dados não utilizadas como selecionadas por NDB1 e NDB2 terão nível baixo e os caracteres recebidos serão normalizados à direita, i.e., o bit menos significativo aparecerá na saída RD1.
13	RPE	Receiver Parity Error	Esta saída tri-state (habilitada por $\overline{SWE}$) fica em alto se o bit de paridade do caracter recebido for diferente da paridade selecionada.
14	RFE	Receiver Framing Error	Esta saída tri-state (habilitada por $\overline{SWE}$) fica em alto se o caracter recebido não tiver um bit de parada válido.
15	ROR	Receiver Over Run	Esta saída tri-state (habilitada por $\overline{SWE}$) fica em alto se o caracter recebido anteriormente não foi lido (RDA = 1) antes que o caracter atual tenha sido transferido para o registro da recepção.
16	$\overline{SWE}$	Status Word Enable	Um nível baixo habilita as saídas (RPE, RFE, ROR, RDA e TBMT) do registro da palavra de estado.
17	RCP	Receiver Clock	Esta entrada é um clock, cuja freqüência é 16 vezes a taxa de baud desejada para recepção.
18	$\overline{RDAR}$	Receiver Data Available Reset	Um nível baixo coloca a saída RDA em baixo.
19	RDA	Receiver Data Available	Esta saída tri-state (habilitada por $\overline{SWE}$) fica em alto quando um caracter inteiro foi recebido e transferido para o registro de recepção.
20	RSI	Receiver Serial Input	Esta entrada aceita uma corrente serial de bits. Uma transição de "1" para "0" é necessária para iniciar a recepção de dados.
21	MR	Master Reset	Esta entrada deve ser pulsada para "1" após ligar a alimentação. Isto colocará em alto TSO, TEOC e TBMT, e em baixo RDA, RPE, RFE e ROR.
22	TBMT	Transmitter Buffer Empty	Esta saída tri-state (habilitada por $\overline{SWE}$) fica em alto quando o registro de transmissão pode ser carregado com novo dado.
23	$\overline{TDS}$	Transmitter Data Strobe	Um nível baixo de Strobe carrega os bits do dado no registro do transmissor.
24	TEOC	Transmitter End of Character	Esta saída fica em alto toda vez que um caracter inteiro é transmitido. Ela fica neste estado até o início de transmissão do próximo caracter ou por meio período TCP no caso de transmissão contínua.
25	TSO	Transmitter Serial Output	Esta saída fornece serialmente o caracter inteiramente transmitido. TSO fica em alto quando não há transmissão de dados.
26-33	TD1-TD8	Transmitter Data Inputs	São 8 linhas de entrada de dados (habilitadas por $\overline{TDS}$). As linhas não utilizadas (selecionadas por NDB1 e NDB2) podem ficar em qualquer estado. O bit menos significativo deve ser sempre colocado em TD1.
34	CS	Control Strobe	Um nível alto nesta linha carrega os bits de controle (NDB1, NDB2, NSB, POE e NPB) no registro de armazenamento dos bits de controle. Esta linha deve ser pulsada ou conectada fisicamente a um nível alto (V_{CC}).

Pino	Sigla	Nome	Função
35	NPB	No Parity Bit	Um nível alto nesta linha elimina o bit de paridade da transmissão. O(s) bit(s) de parada segue(m) imediatamente o último bit de dados. Em compensação, o receptor precisa que o(s) bit(s) da parada venha(m) imediatamente após o último bit de dados. A saída RPE também é forçada a um nível baixo (veja pino 39-POE).
36	NSB	Number of Stop Bits	Esta entrada seleciona o número de bits de parada. Um nível baixo nesta linha seleciona 1 bit de parada. Um nível alto seleciona dois bits de parada. Selecionando-se dois bits de parada, quando se programa uma palavra de dados de apenas 5 bits, gera-se 1,5 bits de parada no caso das COM 2017 ou COM 2017/H.
37-38	NDB2, NDB1	Number of Data Bits/Character	Estas duas entradas são decodificadas internamente para selecionar 5, 6, 7 ou 8 bits de dados/caracter, seguindo a tabela verdade:

NDB2	NDB1	bits de dados/caracter
L	L	5
L	H	6
H	L	7
H	H	8

Obs..
L = baixo
H = alto

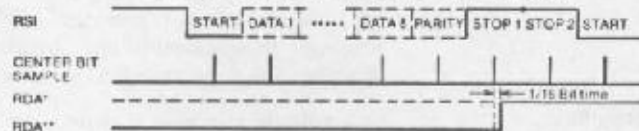
Pino	Sigla	Nome	Função
39	POE	Odd/Even Parity Select	O nível lógico nesta entrada em conjunção com a entrada NPB determina o modo de paridade tanto para o transmissor como para o receptor, seguindo a tabela verdade:

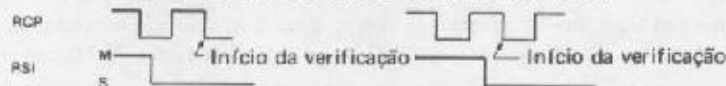
NPB	POE	Modo
L	L	Paridade ímpar
L	H	Paridade par
H	X	Sem paridade

Obs.:
L = baixo
H = alto
X = qualquer nível

Pino	Sigla	Nome	Função
40	TCP	Transmitter Clock	Esta entrada é de um clock cuja freqüência é de 16 vezes (16X) a taxa de baud desejada para transmissão.

No início da transmissão de dados ou quando não se está transmitindo a 100% da utilização da linha, o bit de partida será colocado na linha TSO durante a transição de "1" para "0" do clock TCP, obedecendo a transição negativa de $\overline{TDS}$.

DIAGRAMA DE TEMPO DE RECEPÇÃO
8 BITS, PARIDADE, 2 BITS DE PARADA

* A linha RDA não foi levada previamente a nível baixo (ROR = 1)
** A linha RDA foi levada previamente a nível baixo (ROR = 0)

DETECÇÃO / VERIFICAÇÃO DO BIT DE PARTIDA

Caso a linha RS1 permaneça em baixo pelo tempo de 1/2 bit, um genuíno bit de partida é verificado. Caso esta linha volte a nível alto antes do tempo de 1/2 bit, a verificação do bit de partida é reiniciada.

EED RATINGS* (VALORES MÁXIMOS GARANTIDOS)

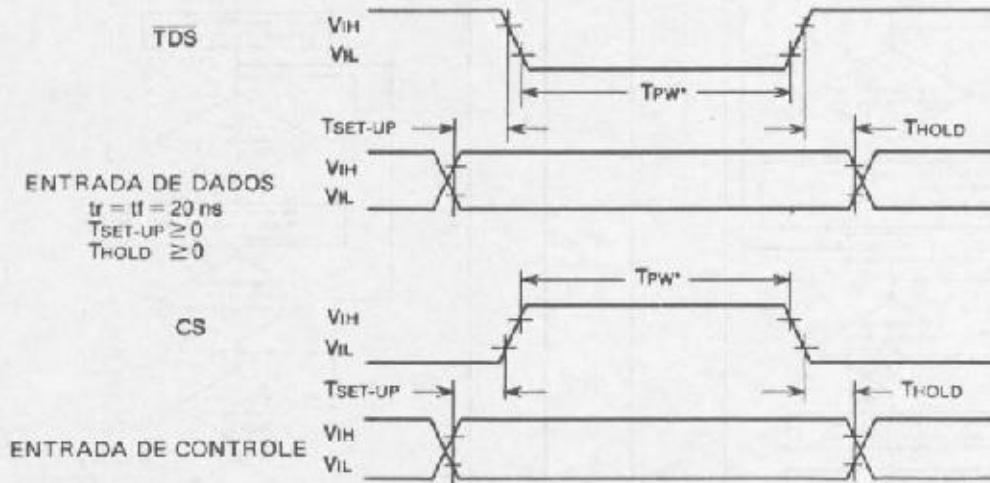
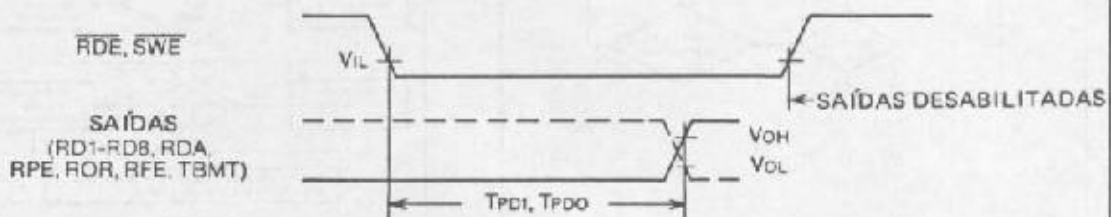
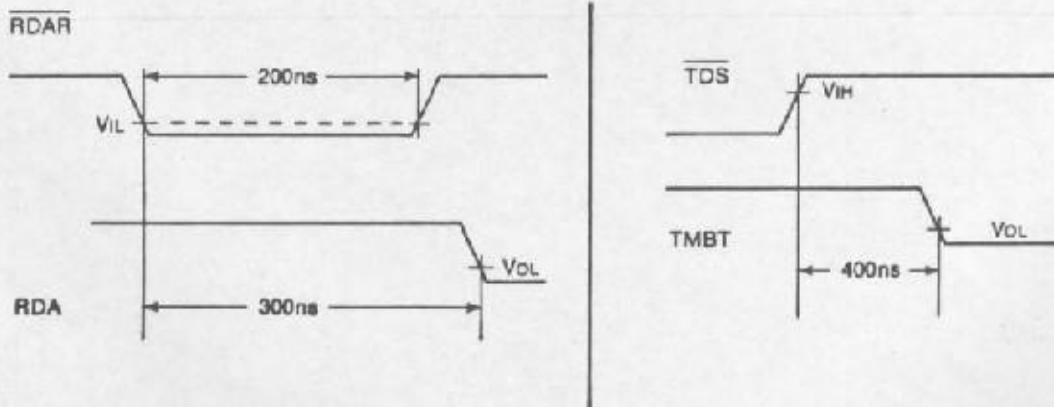
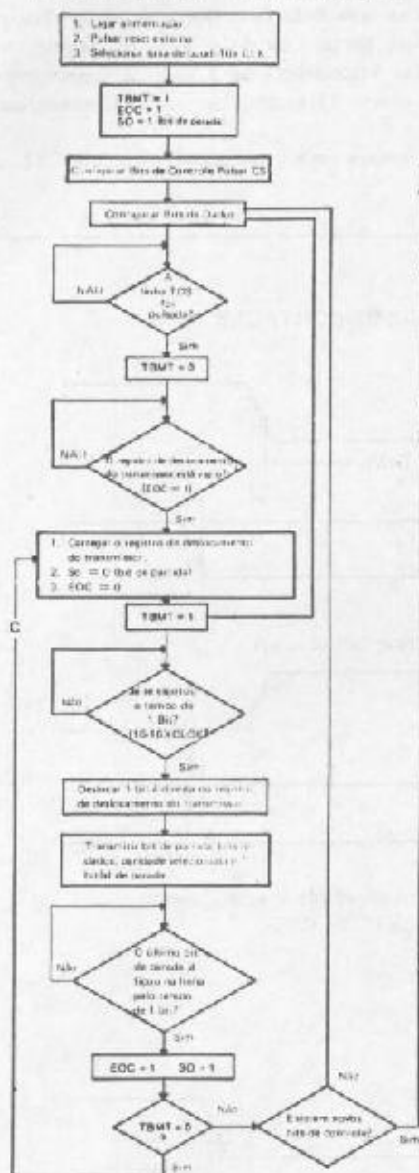
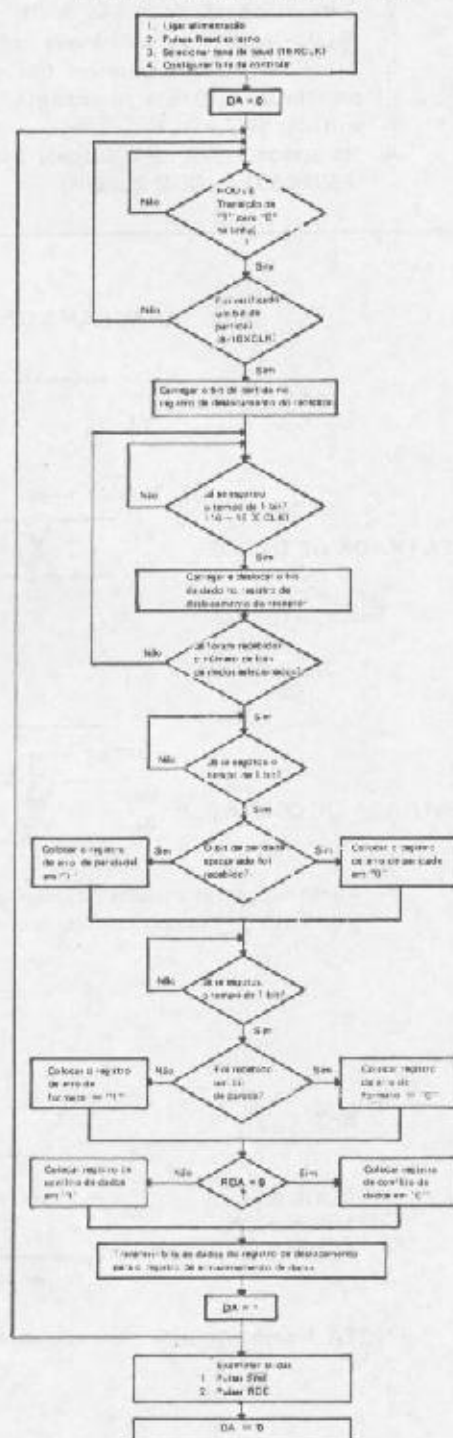
Operating Temperature Range .... 0°C to +70°C
Storage Temperature Range .... −55°C to +150°C
Lead Temperature (soldering, 10 sec.) .... +325°C
Positive Voltage on any Pin, V_{CC} .... +0.3V
Negative Voltage on any Pin, V_{CC} .... −25V

* Comentário – Esforços maiores do que os especificados podem danificar permanentemente o componente. Estes são apenas valores máximos de esforço e a operação funcional do componente nestas condições não está prevista.

CARACTERÍSTICAS ELÉTRICAS (T_A = 0°C a 70°C, V_{CC} = +5V ±5%, V_{DD} = −12V ±5%, a menos que especificado em contrário)

Parâmetro	Mín.	Tipo	Máx.	Unid.	Condições
D.C. CHARACTERISTICS					
INPUT VOLTAGE LEVELS					
Low-level, V_{IL}	V_{DD}		0.8	V	
High-level, V_{IH}	V_{CC}-1.5		V_{CC}	V	
OUTPUT VOLTAGE LEVELS					
Low-level, V_{OL}		0.2	0.4	V	I_{OL} = 1.6mA
High-level, V_{OH}	2.4	4.0		V	I_{OH} = 100µA
INPUT CURRENT					
Low-level, I_{IL}			1.6	mA	see note 4
OUTPUT CURRENT					
Leakage, I_{LO}			−1	µA	$\overline{SWE} = \overline{RDE} = V_{IH}$, $0 \leq V_{OUT} \leq +5V$
Short circuit, I_{OS}**			10	mA	V_{OUT} = 0V
INPUT CAPACITANCE					
All inputs, C_{IN}		5	10	pf	$V_{IN} = V_{CC}$, f = 1MHz
OUTPUT CAPACITANCE					
All outputs, C_{OUT}		10	20	pf	$\overline{SWE} = \overline{RDE} = V_{IH}$, f = 1MHz
POWER SUPPLY CURRENT					
I_{CC}			28	mA	All outputs = V_{OH}, All inputs = V_{CC}
I_{DD}			28	mA	
A.C. CHARACTERISTICS					T_A = +25°C
CLOCK FREQUENCY					
(COM2502, COM2017)	DC		400	KHz	RCP, TCP
(COM2502H, COM2017H)	DC		640	KHz	RCP, TCP
PULSE WIDTH					
Clock	1			µs	RCP, TCP
Master reset	500			ns	MR
Control strobe	200			ns	CS
Transmitter data strobe	200			ns	$\overline{TDS}$
Receiver data available reset	200			ns	$\overline{RDAR}$
INPUT SET-UP TIME					
Data bits	≥0			ns	TD1-TD8
Control bits	≥0			ns	NPB, NSB, NDB2, NDB1, POE
INPUT HOLD TIME					
Data bits	≥0			ns	TD1-TD8
Control bits	≥0			ns	NPB, NSB, NDB2, NDB1, POE
STROBE TO OUTPUT DELAY					Load = 20pf +1 TTL input
Receive data enable			350	ns	$\overline{RDE}$: T_{PD1}, T_{PD0}
Status word enable			350	ns	$\overline{SWE}$: T_{PD1}, T_{PD0}
OUTPUT DISABLE DELAY			350	ns	$\overline{RDE}$, $\overline{SWE}$

** Não mais que uma saída deve ser curtocircuitada de uma vez.

NOTAS: 1. Se o transmissor estiver inativo (TEOC e TBMT em alto), o bit de partida aparecerá na linha TSO pelo espaço de um período de clock (TCP) após a transição negativa de $\overline{TDS}$.
2. O bit de partida (transição de "1" para "0") será sempre detectado durante um período de clock (RCP), garantindo um desvio máximo do bit de partida de 1/16 avos do tempo de 1 bit.
3. As saídas tri-state possuem três estados: 1) baixa impedância para V_{CC}; 2) baixa impedância para terra, e 3) alta impedância OFF ≅ 10M ohms. O estado de "OF" é controlado pelas entradas $\overline{SWE}$ e $\overline{RDE}$.
4. No estado estacionário (Steady State) não flui corrente para interfaceamento com TTL ou MOS (COM 2502 ou COM 2502/H).

FLUXOGRAMA – TRANSMISSOR
FLUXOGRAMA – RECEPTOR
TBMT = 0
TBMT = 1
EOC = 1 SO = 1
TBMT = 0
DA = 0
DA = 1
RDA = 0
Sim
Não

APÊNDICE C8

CRT 5027
CRT 5037
CRT 5057*

CRT Vídeo Timer and Controller

UTAC

(Controlador e Temporizador de Vídeo CRT)
Obs.: CRT — tubo de raios catódicos

FACILIDADES

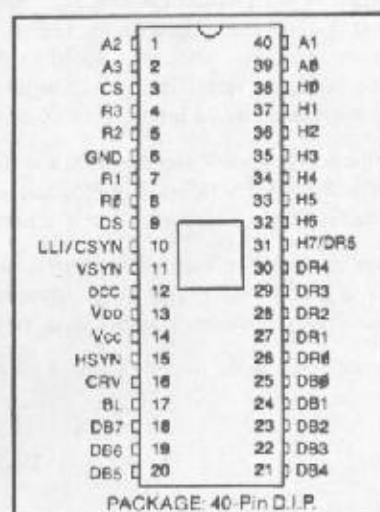
- Formato do Display totalmente programável
  - Caracteres por linha (1 a 200)
  - Linhas de dados por tela (1 a 64)
  - Varreduras por linha (1 a 16)

- Formato programável do sincronismo do monitor
  - Varreduras/tela (256 a 1023)
  - Parte visível "front porch"
  - Largura de SYNC
  - Parte não visível "back porch"
  - Interligação/Não-interligação
  - Apagamento vertical

- Entrada Lock Line (CRT 5057)

- Saídas diretas para o monitor de vídeo
  - Sync horizontal
  - Sync vertical
  - Sync composto (CRT 5057, CRT 5037)
  - Apagamento
  - Cursor coincidente

- Programação através de:
  - Barra de dados do sistema
  - PROM externa
  - ROM de máscara opcional

CONFIGURAÇÃO DOS PINOS

- Compatível com CRT standard ou não-standard

- Taxa de restauração (Refresh): 50 Hz, 60 Hz

- Rolamento
  - Por linha
  - Multilinhas

- **Registro de posicionamento do cursor**
- **Formato do caracter: 5 × 67, 7 × 9 ...**
- **Posicionamento vertical de dados programável**
- **Interligação da corrente de feixe, balanceado (CRT 5037)**
- **Compatibilidade gráfica**
- **Aplicações de Split-Screen**
  - **Vertical**
  - **Horizontal**
- **Operação Interligada/Não-interligada**
- **Compatibilidade TTL**
- **Projetado para barras**
- **Operação em alta velocidade**
- **Tecnologia de canal-N de silício COPLAMOS**
- **Compatível com CRT 8002 VDAC**
- **Compatível com CRT 7004**

DESCRIÇÃO GERAL

A pastilha controladora e temporizadora de CRT de vídeo (VTAC) é um componente programável de 40 pinos MOS/LSI, canal-N, COPLAMOS que contém as funções lógicas necessárias para gerar todos os sinais de temporização para apresentar e formatar dados de vídeo, interligamento ou não para monitores de CRT standard ou não.

Com exceção do contador de pontos, que deve ser sincronizado com uma freqüência de vídeo acima de 25 MHz e, portanto, não recomendado para implementação com MOS, toda formatação de tela, tais como sincronismos horizontal, vertical e composto, caracteres por linha, linhas por tela e varreduras por linha de caracteres e por tela, é totalmente programável. O contador de linha de caracteres foi projetado para facilitar o rolamento.

A programação é feita através da carga de 7 registros de controle de 8 bits diretamente de uma barra bidirecional de dados de 8 bits. Quatro registros de linhas de endereçamento e uma linha de habilitação da pastilha proporcionam uma compatibilidade total com microprocessadores, com respeito à configuração inicial de programação. A pastilha pode ser "autocarregada" por uma PROM externa, colocada na barra de dados, como explicado na seção de OPERAÇÃO. A formatação também pode ser programada através da opção de mascaramento.

Além dos sete registros de controle, existem dois registros adicionais para armazenar os endereços do cursor da linha de caracteres, para geração do sinal de vídeo com cursor. O conteúdo destes dois registros pode ser lido também pela barra para atualização por programa.

Existem três versões do VTAC. O CRT 5027 fornece operação não interligada com um número par ou ímpar de varreduras por linha de caracteres, ou operação interligada com número par de varreduras por linha de caracteres. O CRT 5037 pode ser programado para um número par ou ímpar de varreduras por linha de caracteres em ambos os modos, interligado ou não. Programando-se o CRT 5037 para um número ímpar de varreduras por linha de caracteres, elimina-se a distorção dos caracteres causada por uma corrente de feixe não-par, normalmente associada à interligação de campo ímpar/campo par de displays alfanuméricos.

O CRT 5057 fornece a capacidade de se "amarrar" (Lock) à taxa de restauração vertical do CRT, como controlada pelo pulso de sincronismo vertical do VTAC com a freqüência da rede de 50 Hz ou 60 Hz, eliminando-se assim o efeito conhecido por "Swim" (ondulação). Esta facilidade é especialmente útil para os sistemas europeus.

A forma de onda de freqüência da rede, transformada para os níveis lógicos requeridos pelo VTAC, é aplicada na entrada de Lock. O VTAC irá inibir a geração de um pulso de sincronismo vertical até ocorrer uma transição de "0" para "1" nesta entrada. O pulso de Sync vertical é, então, iniciado durante uma varredura de linha após esta transição passar pelo limite de estado "1" do VTAC.

Para possibilitar a existência do pino de Lock, o pino de sincronismo composto não é fornecido no CRT 5057.

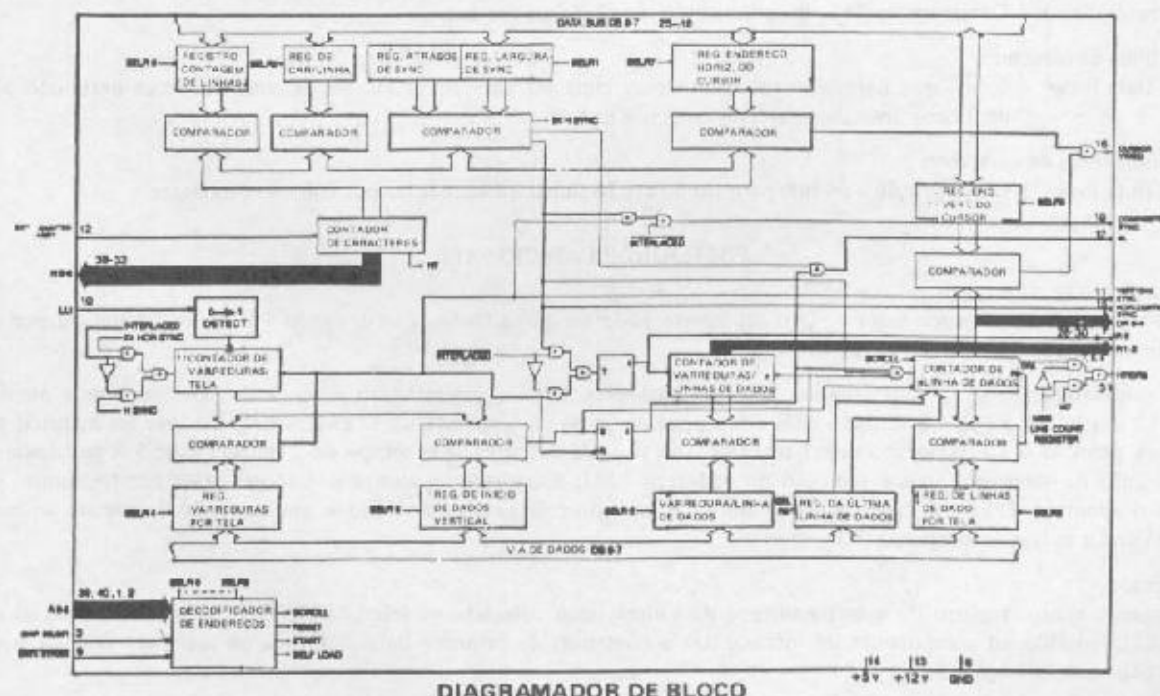
DESCRIÇÃO FUNCIONAL DOS PINOS

Nº PINO	SÍMBOLO	NOME	ENTRADA/ SAÍDA	FUNÇÃO
25-18	DB0-7	Data Bus	E/S	Barra de dados. Barra de entrada para palavras de controle do microprocessador ou PROM. Barra bidirecional para endereço do cursor.
3	CS	Chip Select	E	Sinal de seleção da pastilha.
39, 40, 1, 2	A0-3	Register Address	E	Bits de endereçamento para selecionar um entre os sete registros de controle ou os registros de endereços do cursor.
9	$\overline{DS}$	Data Strobe	E	Armazena DB0-7 no registro apropriado ou libera o endereço do cursor ou da linha do cursor para a barra de dados.

12	DCC	DOT Counter Carry	E	Transbordo do contador de pontos externo, estabelecendo taxa básica para caracter. Clock do caracter.
38-32	H0-6	Character Counter Output	S	Saídas do contador de caracteres.
7, 5, 4	R1-3	Scan Counter LSB	S	Três bits mais significativos do contador de varredura; entradas de seleção de linha para o gerador de caracteres.
31	H7/DR5	H7/DR5	S	A definição deste pino é programável. A saída será o MSB (bit mais significativo) do contador de caracteres se o contador de linhas horizontais (REG0) for ⩾ 128; caso contrário, a saída será o MSB do contador de linha de caracter.
8	R0	Scan Counter LSB	S	Bit menos significativo do contador de varreduras. No modo interligado com um número par de varreduras por linha de caracter, R0 irá pulsar na taxa do campo; para um número ímpar de varreduras por linha de caracter no modo interligado, R0 irá pulsar na taxa da linha de caracteres.
26-30	DR0-4	Data Row Counter Outputs	S	Saídas do contador de linha de caracteres.
17	BL	Blank	S	Define a porção não ativa das varreduras horizontais e verticais.
15	HSYN	Horizontal Sync	S	Início do retraço horizontal.
11	VSYN	Vertical Sync	S	Início do retraço vertical.
10	CSYN/ LLI	Composite Sync Output/ Line Lock Input	S/E	Nos CRT 5027 e CRT 5037 são fornecidos um sinal de sincronismo composto. Esta saída fica ativa apenas no modo não interligado. É fornecida uma forma de onda real de sincronismo composto do tipo RS-170. Para o CRT 5057 este pino é para entrada de Lock com a rede.
16	CRV	Cursor Video	S	Define a localização do cursor do campo de dados.
14	V_{CC}	Power Supply	F.A.	Fonte de alimentação de +5V.
13	V_{DD}	Power Supply	F.A.	Fonte de alimentação de +12V.

OPERAÇÃO

A filosofia de projeto empregada foi a de permitir o componente interfacear tanto com um sistema baseado em microprocessador como em sistemas de lógica discreta. O componente pode ser programado de duas maneiras pelo usuário: através da barra de dados do processador como parte da rotina de inicialização, ou ao ligar através de uma PROM colocada na barra de dados e endereçada

DIAGRAMADOR DE BLOCO

diretamente pelas saídas de seleção de linha da pastilha (vide figura 4). Sete palavras de 8 bits são necessárias para a completa programação da pastilha. A configuração dos bits destas palavras está mostrada na Tabela 1. A informação contida nestas sete palavras consiste no seguinte:

Formatação Horizontal:
Caracteres/linha
(Characters/Data Row) – Código de 3 bits fornecendo 8 tamanhos possíveis da linha de caracteres de 20 a 132. Um componente standard poderá ser configurado com linhas de 20, 32, 40, 64, 72, 80, 96 e 132 caracteres.

Retardo de sincronismo horizontal
(Horiz. Sync Delay) – Configuração de 3 bits fornecendo até 8 tempos de caracter para geração da "Parte Visível" (front porch).

Largura do sincronismo horizontal
(Horiz. Sync Width) – Configuração de 4 bits fornecendo até 15 tempos de caracter para geração da largura do sincronismo horizontal.

Contagem de linhas horizontais
(Horiz. Line Count) – Configuração de 8 bits fornecendo até 256 tempos de caracter para uma total formatação horizontal.

Bits de atraso
(Skew bits) – Código de 2 bits, fornecendo atraso de 0 a 2 caracteres entre o contador de endereço horizontal e os sinais de Blank e de Sync (Horiz. Vert. e composto) para permitir a retemporização dos dados de vídeo antes da geração do sinal de vídeo composto. O sinal de vídeo de cursor também é atrasado em função deste código.

Formatação Vertical:
Interligado/Não-interligado
(Interlaced/
/Non interlaced) – Este bit fornece apresentação de dados com formatação do campo ímpar/par para sistemas interligados. O tempo dos contadores verticais é modificado como descrito abaixo. Um nível alto estabelece o modo interligado.

Varredura/Tela
(Scans/Frame) – Configuração de 8 bits definidos de acordo com as seguintes equações: seja X = valor da configuração dos 8 bits.

1) No modo interligado – Varreduras/Tela = 2X + 513. Então para 525 varreduras deve-se programar X = 6 (00000110). O sincronismo vertical ocorrerá precisamente a cada 262,5 varreduras, produzindo, assim, dois campos interligados.
   Alcance = 513 a 1023 varreduras/Tela, apenas contagens ímpares.
2) No modo não-interligado – Varreduras/tela = 2X + 256. Então para 262 varreduras deve-se programar X = 3 (00000011)
   Alcance = 256 a 766 varredura/tela, apenas contagens pares. Em ambos os modos, a largura de sincronismo vertical é fixada em três varreduras horizontais (= 3H).

Início de dados vertical
(Vert. Data Start) – 8 bits definindo o número de linhas de varreduras existentes do início do pulso de Sync vertical até o início do pulso de Sync vertical até o início do aparecimento de dados na tela. Nesta linha de varredura, o contador de linha de caracteres é configurado com o endereço da linha de caracteres do início da tela.

Linhas de caracteres/tela
(Data Rows/Frame) – Configuração de 6 bits permitindo até 64 linhas por tela.

Última linha de caracter
(Last Data Row) – 6 bits que permitem rolamento para cima ou para baixo através de uma pré-carga definindo a contagem da última linha de caracteres colocada na tela.

Varreduras/linhas de caracteres
(Scan/Data Row) – Configuração de 4 bits permitindo até 16 linhas de varreduras por linha de caracteres.

FACILIDADES ADICIONAIS

Inicialização do componente:

Sob controle de microprocessador – O componente pode ser inicializado com o estado "0" pelo sistema ou por programa, colocando-se o endereço 1010 nas linhas A3-0.

Por "autocarregamento" – Em sistemas sem processadores, o autocarregamento é realizado colocando-se e mantendo-se o endereço 1111 nas linhas A3-0, e é iniciado com a recepção do pulso de Strobe ($\overline{DS}$). O endereço 1111 deve ser mantido por tempo suficiente para permitir o carregamento dos 7 registros. (Na maioria das aplicações tempo de 1 milissegundo.) A seqüência de tempo iniciará uma linha de varredura após a remoção do endereço 1111. Em sistemas com processadores, o autocarregamento tem início colocando-se o endereço 0111 nas linhas A3-0. O autocarregamento é finalizado enviando-se um comando de partida ao componente que também inicia a cadeia de temporização.

Rolamento:

Juntamente com o registro de armazenamento da última linha colocada na tela (REGISTRO 6), um comando de rolamento (endereço 1011) enviado ao componente irá incrementar a contagem da primeira linha colocada na tela para facilitar o rolamento para cima em algumas aplicações.

MAPA DE PROGRAMAÇÃO DOS REGISTROS DE CONTROLE

Contagem de linhas horizontais: total de caracteres/linha = N + 1, N = 0 a 255 (DEBO = LSB).
Obs.: LSB = bit menos significativo.

Caracteres/linha:

DB2	DB1	DB0		
0	0	0	=	20 caracteres ativos/linha
0	0	1	=	32
0	1	0	=	40
0	1	1	=	64
1	0	0	=	72
1	0	1	=	80
1	1	0	=	96
1	1	1	=	132

Retardo do sincronismo horizontal: N, de 1 a 7 tempos de caracter (DBO = LSB) (N = 0 não é permitido).

Largura do sincronismo horizontal: = N, de 1 a 15 tempos de caracter (DB3 = LSB) (N = não é permitido).

Bits de atraso:

DB7	DB8	atraso SYNC/BCANC (tempos de caracteres)	atraso do cursor
0	0	0	0
1	0	1	0
0	1	2	1
1	1	2	2

Varreduras/tela: Configuração de 8 bits definidos de acordo com as seguintes equações: seja X = valor da configuração dos 8 bits.

1) No modo interligado – Varreduras/tela = 2X + 513. Então para 525 varreduras deve-se programar X = 6 (00000110). O sincronismo vertical ocorrerá precisamente a cada 262,5 varreduras produzindo, assim, dois campos interligados.
   Alcance = 513 a 1023 varreduras/tela, apenas contagens ímpares.
2) No modo não interligado – Varreduras/tela = 2X + 256. Então para 262 varreduras deve-se programar X = 3 (00000011).
   Alcance = 256 a 766 varredura/tela, apenas contagens pares.
   Em ambos os modos, a largura do sincronismo vertical é fixada em três varreduras horizontais (= 3H).

Início de dados vertical: N = número de atraso de linhas da varredura após a transição do sincronismo vertical da posição inicial do vertical (DBO = LSB).

Linhas caracteres/tela: Número de linhas de caracteres = N + 1, N = 0 a 63 (DBO = LSB).

Última linha de caracteres: N = endereço da última linha de caracteres colocada no vídeo, N = 0 a 63, ex.; para 24 linhas de caracteres, programas N = 23. (DBO = LSB).

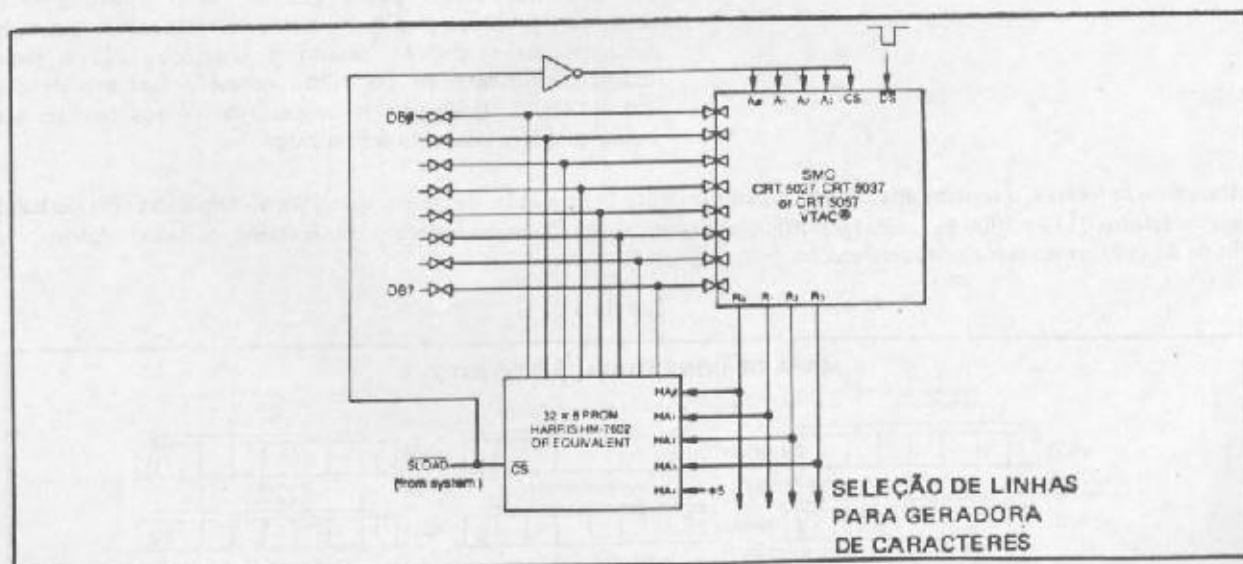



Figura 4 Esquema de autocarregamento para configuração do VTAC.

Modo:	No registro 1, DB7 = 1 estabelece modo de interligação.
Varreduras/linha de caracteres:	[Modo interligado] CRT 5027: Varredura/linha de caracteres = N + 1, onde N = número programado de linha de caracteres. N = 0 a 15. As varreduras por linha de caracteres devem ser em número par. CRT 5037, CRT 5057: Varreduras/linhas de caracteres = N + 2. N = 0 a 14, números pares ou ímpares. [Modo não interligado] CRT 5027, CRT 5037, CRT 5057: Varreduras/linha de caracteres = N + 1. N = 0 a 15, números pares ou ímpares.

Seleção de registros/Códigos de comando

A3	A2	A1	A0	Seleção/Comando	Descrição
0	0	0	0	Carregar registro 0	
0	0	0	1	Carregar registro 1	
0	0	1	0	Carregar registro 2	
0	0	1	1	Carregar registro 3	
0	1	0	0	Carregar registro 4	Veja Tabela 1
0	1	0	1	Carregar registro 5	
0	1	1	0	Carregar registro 6	
0	1	1	1	Processador inicia a autocarga	Comando do processador instruindo o VTAC para entrar em modo de autocarga (via PROM externa)
1	0	0	0	Ler endereço da linha do cursor	
1	0	0	1	Ler endereço do cursor	
1	0	1	0	Reset	Inicializa a cadeia de temporização para o canto superior esquerdo da tela. O reset é dado pela linha $\overline{DS}$ e os contadores são presos até serem liberados pelo comando de partida.
1	0	1	1	Rolamento para cima	Incrementa o endereço da primeira linha de caracteres colocada na tela ex, antes de receber o comando de rolamento – linha superior = 0, linha inferior = 23. Após a recepção deste comando – linha superior = 1, linha inferior = 0.
1	1	0	0	Carregar o endereço do cursor*	
1	1	0	1	Carregar o endereço da linha do cursor*	
1	1	1	0	Iniciar cadeia de temporização	A recepção deste comando após um Reset ou um comando de autocarga irá disparar a cadeia de temporização após uma linha de varredura aproximadamente. Em aplicações que requerem operação síncrona de mais de um CRT 5027, o transbordo do contador de pontos deve ser mantido baixo durante o $\overline{DS}$ para este comando.
1	1	1	1	Autocarga sem processador	O componente começará a autocarga via PROM quando $\overline{DS}$ ficar baixo. O comando 1111 deve ser mantido na linha de endereço A3-0 por tempo suficiente para garantir a autocarga (o contador de varreduras deverá passar por um ciclo inteiro pelo menos uma vez). A autocarga é terminada automaticamente e a cadeia de temporização é iniciada quando o endereço 1111 é removido, independentemente de $\overline{DS}$. Para operação síncrona de mais de um VTAC, o transbordo do contador de pontos deve ser mantido baixo quando o comando for retirado.

*NOTA: Durante a autocarga, o registro de endereço do cursor (REG 7) e o registro de endereço da linha do cursor (REG 8) são habilitados durante os estados 0111 e 1000 das saídas R3-R0 do contador de varreduras respectivamente. Portanto, os dados relativos ao cursor na PROM deverão ser gravados nestes endereços.

Tabela 1

DIAGRAMA DE TEMPO CA

Figura 1 Temporização de vídeo

CONTADOR DE PONTO

PW_L PW_H

H∅-7 H SYNC, V SYNC, BLANK CURSOR VIDEO COMPOSITE SYNC

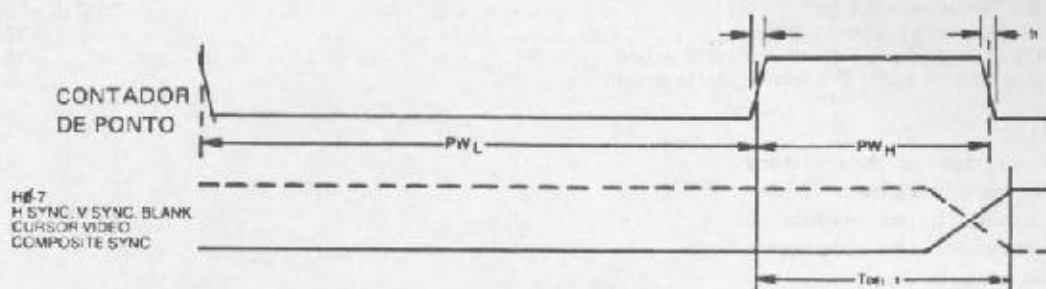



Figura 2 Temporização de carga/leitura

ENDEREÇO SELEÇÃO DA PASTILHA

DB∅-7 CARREGAMENTO DOS DADOS

DB∅-7 LEITURA DOS DADOS

$\overline{DS}$

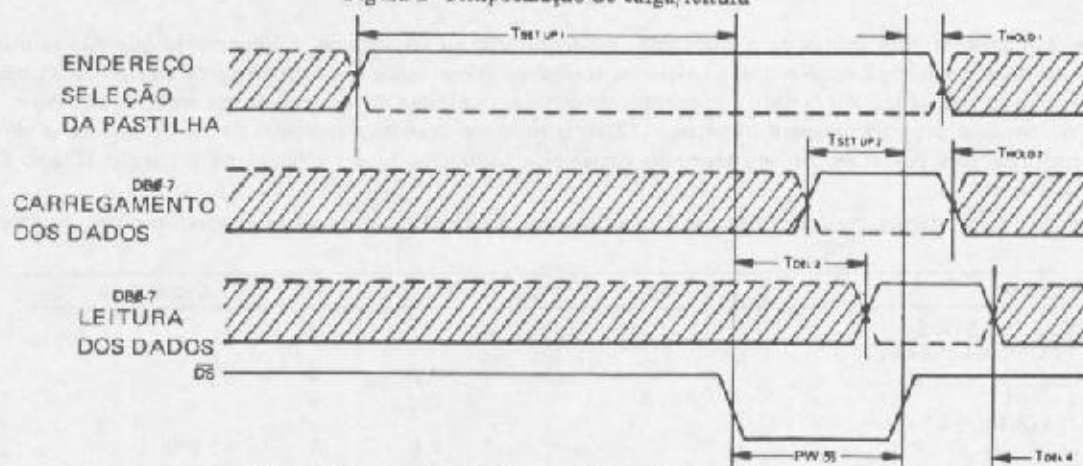
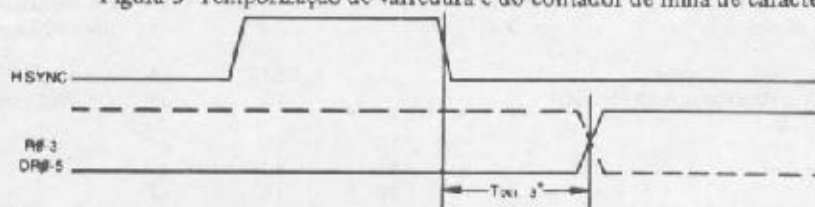
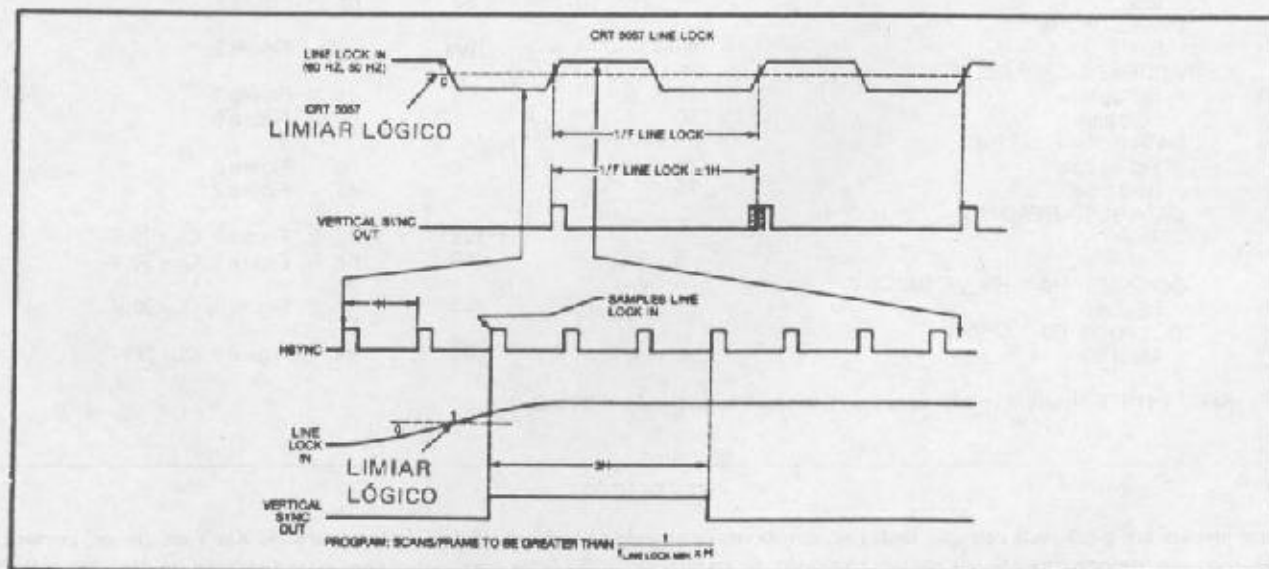



Figura 3 Temporização de varredura e do contador de linha de caracteres

HSYNC

R∅-3 DR∅-5

* R0-3 e DR0-5 devem mudar antes da transição negativa do Sync horizontal.

PROGRAMAÇÃO: VARREDURAS/TELA PARA SER MAIOR QUE $\frac{1}{f_{LINE\ LOCK\ MÍN} \times H}$

Maximum Guaranteed Ratings* (Valores Máximos Garantidos)

Operating Temperature Range .... 0°C to + 70°C
Storage Temperature Range .... −55°C to +150°C
Lead Temperature (soldering, 10 sec) .... +325°C
Positive Voltage on any Pin, with respect to ground .... +18.0V
Negative Voltage on any Pin, with respect to ground .... −0.3V

* COMENTÁRIO
Esforços maiores do que os especificados podem danificar permanentemente o componente. Estes são apenas valores máximos de esforço, e a operação funcional do componente nestas condições não está prevista.

NOTA:
Quando alimentar este componente com fontes de alimentação de laboratório ou do sistema, é importante que não se ultrapassem os Valores Máximos Absolutos ou poderá ocorrer falha. Algumas fontes de alimentação apresentam perturbações nas saídas (spikes, glitches) quando são ligadas ou desligadas. Além disto, transientes de voltagem na rede CA podem aparecer na saída CC. Por exemplo, a fonte de alimentação de bancada programada para fornecer + 12 volts pode ter grandes transientes de tensão quando a alimentação CA é ligada ou desligada. Se isto por acaso estiver existindo sugere-se a utilização de um circuito de limitação (Clamp Circuito).

CARACTERÍSTICAS ELÉTRICAS (T_A=0°C to 70°C, V_{CC}= +5V±5%, V_{DD}= +12V±5%, a menos que especificado em contrário).

Parameter	Min.	Typ.	Max.	Unit	Comments
D.C. CHARACTERISTICS					
INPUT VOLTAGE LEVELS					
Low Level, V_{IL}			0.8	V	
High Level, V_{IH}	V_{CC}−1.5		V_{CC}	V	
OUTPUT VOLTAGE LEVELS					
Low Level—V_{OL} for RØ-3			0.4	V	I_{OL} = 3.2ma
Low Level—V_{OL} all others			0.4	V	I_{OL} = 1.6ma
High Level—V_{OH} for RØ-3, DBØ-7	2.4				I_{OH} = 80μa
High Level—V_{OH} all others	2.4				I_{OH} = 40μa
INPUT CURRENT					
Low Level, I_{IL} (Address, CS only)			250	μA	V_{IL} = 0.4V
Leakage, I_{IL} (All Inputs except Address, CS)			10	μA	$0 \leq V_{IN} \leq V_{CC}$
INPUT CAPACITANCE					
Data Bus, C_{IN}		10	15	pF	
$\overline{DS}$, Clock, C_{IN}		25	40	pF	
All other, C_{IN}		10	15	pF	
DATA BUS LEAKAGE in INPUT MODE					
I_{DB}			10	μA	$0.4V \leq V_{IN} \leq 5.25V$
POWER SUPPLY CURRENT					
I_{CC}		80	100	mA	
I_{DD}		40	70	mA	
A.C. CHARACTERISTICS					T_A = 25°C
DOT COUNTER CARRY					
frequency	0.2		4.0	MHz	Figure 1
PW_H	35			ns	Figure 1
PW_L	215			ns	Figure 1
t_r, t_f		10	50	ns	Figure 1
DATA STROBE					
PW_{DS}	150ns		10μs		Figure 2
ADDRESS, CHIP SELECT					
Set-up time	125			ns	Figure 2
Hold time	50			ns	Figure 2
DATA BUS—LOADING					
Set-up time	125			ns	Figure 2
Hold time	75			ns	Figure 2
DATA BUS—READING					
T_{DEL2}			125	ns	Figure 2, CL=50pF
T_{DEL4}	5		60	ns	Figure 2, CL=50pF
OUTPUTS: HØ-7, HS, VS, BL, CRV, CS-T_{DEL1}			125	ns	Figure 1, CL=20pF
OUTPUTS: RØ-3, DRØ-5 T_{DEL3}	*		500	ns	Figure 3, CL=20pF

* RO-3 e DRO-5 devem mudar antes da transição negativa de H SYNC.

RESTRIÇÕES

1. Existe apenas um pino para carregar dados no componente através da barra de dados. As coordenadas X e Y do cursor, portanto, são carregadas no componente através de um conjunto de endereços, e são lidas com outro conjunto diferente de endereços. Portanto, os sinais de WRITE e READ (escritura e leitura), padrão na maioria dos microprocessadores, deverão passar por uma porta NOR externa para gerar um único sinal de strobe ($\overline{DS}$) para o componente.
2. No modo interligado, o número total de espaços para caracteres configurados para varredura horizontal deverá ser par para assegurar que o sincronismo vertical ocorra precisamente entre os pulsos de sincronismo horizontal.

Inicialização, CRT 5027

Quando se faz o carregamento dos registros do CRT 5027 controlado por microprocessador, é necessária a seguinte seqüência de instruções:

ENDEREÇO				COMANDO
1	1	1	0	Iniciar cadeia de temporização
1	0	1	0	Reset
0	0	0	0	Carregar registro 0
	•			•
	•			•
	•			•
0	1	1	0	Carregar registro 6
1	1	1	0	Iniciar cadeia de temporização

A seqüência de INICIAR, RESET, CARREGAR, INICIAR é necessária para assegurar a inicialização correta dos registros. Esta seqüência não é necessária quando se utilizam os modos de autocarga dos registros. Esta seqüência é opcional no caso do CRT 5037 ou CRT 5057.

APÊNDICE C9

CRT 8002

CRT VIDEO DISPLAY ATTRIBUTES CONTROLLER

VIDEO GENERATOR

VDAC

(Controlador de Atributos de Vídeos CRT e Gerador de Vídeo)

FACILIDADES

- Geradora de caracteres interna (mascarável)
  128 caracteres (alfanumérico e gráfico) matriz
  de ponto 7 X 11

CONFIGURAÇÃO DOS PINOS

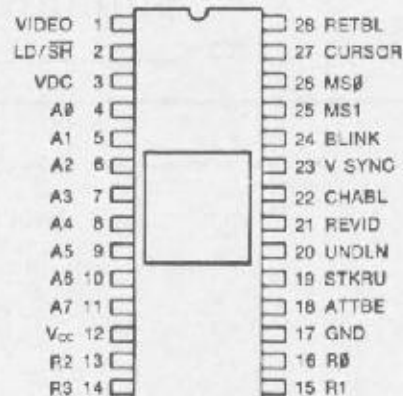
Sinal	Pino	Pino	Sinal
VIDEO	1	28	RETBL
LD/$\overline{SH}$	2	27	CURSOR
VDC	3	26	MSØ
AØ	4	25	MS1
A1	5	24	BLINK
A2	6	23	V SYNC
A3	7	22	CHABL
A4	8	21	REVID
A5	9	20	UNDLN
A6	10	19	STKRU
A7	11	18	ATTBE
V_{cc}	12	17	GND
R2	13	16	RØ
R3	14	15	R1

- Registro de deslocamento de vídeo interno
  Freqüência de deslocamento máxima
  CRT 8002A 20 MHz
  CRT 8002B 10 MHz
  CRT 8002C 10 MHz
  Tempo de acesso 400 ns

- Apagamento de vídeo de retraço horizontal
  e vertical interno

- Não requer circuitos complementares

- 4 modos de operação (intermixáveis)
  Geradora de caracteres interna (ROM)
  Gráficos espessos
  Gráficos finos
  Entradas externas
  (fontes/gráficos de pontos)

- Atributos internos – caracter, campo vídeo reverso
  Apagamento de caracter
  Caracter piscando
  Sublinhado
  Superposição

- 4 modos de cursor
  sublinhado
  sublinhado piscando
  vídeo reverso
  vídeo reverso piscando

- Taxa programável de piscagem do caracter

- Taxa programável de piscagem do cursor

- Subscritível

- **Conjunto expansível de caracteres**
  Fontes externas
  Alfanuméricos e gráficos
  RAM, ROM e PROM
- **Buffer de endereços internos**
- **Buffer de atributo interno**
- **Operação em +5V**
- **Compatível com TTL**
- **Processo COPLAMOS de canal-N porta de silicone MOS**
- **Tecnologia CLASP – ROM e opções**
- **Compatível com CRT 5027 VTAC**

DESCRIÇÃO GERAL

O SMC CRT 8002 VDAC (controlador de atributos de vídeo) é um componente COPLAMOS – MOS/LSI canal-N que utiliza tecnologia CLASP. Ele contém uma ROM geradora de caracteres 7X11X128, um modo de gráficos espessos, modo de gráficos finos, modo de entrada externa, armazenador de dados/endereço de caracteres, lógica de atributo de campo e/ou de caracteres, armazenador de atributo, quatro modos de cursor, 2 taxas programáveis de piscagem do cursor e um registro de deslocamento de vídeo de alta velocidade. O CRT 8002 VDAC é uma pastilha conjugada ao SMC CRT 5027 VTAC. Juntas, estas duas pastilhas desempenham a função de todo circuito para a parte de "display" (mostrador) de um terminal de vídeo CRT.

A saída de vídeo do CRT 8002 pode ser conectada diretamente à entrada do monitor de vídeo CRT. A saída de apagamento do CRT 5027 pode ser conectada diretamente à entrada de apagamento de retraço do CRT 8002 para fornecer um apagamento de retraços vertical e horizontal à saída de vídeo.

Existem quatro modos de cursor no CRT 8002. São eles: sublinhado, sublinhado piscando, vídeo reverso e vídeo reverso piscando. Qualquer um destes modos pode ser programado por mascaramento como uma função do cursor. Existe uma taxa de piscagem do cursor separada que pode ser programada para fornecer uma taxa de 15 Hz a 1 Hz para piscagem.

Os atributos do CRT 8002 incluem: vídeo reverso, apagamento de caracter, piscagem, sublinhamento e superposição. A taxa de piscagem de caracteres pode ser programável por mascaramento de 7,5 Hz a 0,5 Hz com um ciclo de trabalho de 75/25. O sublinhamento e a superposição são funções similares, mas controladas independentemente, e são programáveis por mascaramento para qualquer número de linhas de varredura no bloco do caracter. Estes atributos podem ser utilizados em todos os modos.

No modo de gráfico amplo, o CRT 8002 produz uma entidade gráfica do tamanho de um bloco de caracteres. A entidade gráfica contém 8 partes, cada uma delas associada a um bit de um byte gráfico, formando assim 256 símbolos gráficos únicos. Sendo assim, o CRT 8002 pode produzir tanto um símbolo alfanumérico como uma entidade gráfica, dependendo do modo selecionado. O modo pode ser trocado através de uma base de caracteres.

O modo de gráfico fino permite ao usuário estender o conjunto de caracteres da ROM interna e/ou as capacidades gráficas internas, inserindo símbolos externos. Estes símbolos externos podem chegar via RAM, ROM ou PROM.

CARACTERÍSTICAS ELÉTRICAS (T_A=0°C a 70°C, Vcc=+5V±5%, a menos que especificado em contrário)

Parameter	Min.	Typ.	Max.	Unit	Comments
D.C. CHARACTERISTICS					
INPUT VOLTAGE LEVELS					
Low-level, V_{IL}			0.8	V	excluding VDC
High-level, V_{IH}	2.0			V	excluding VDC
INPUT VOLTAGE LEVELS-CLOCK					
Low-level, V_{IL}			0.8	V	
High-level, V_{IH}	4.3			V	See Figure 6
OUTPUT VOLTAGE LEVELS					
Low-level, V_{OL}			0.4	V	I_{OL} = 0.4 mA, 74LSXX load
High-level, V_{OH}	2.4			V	$I_{OH} = -20\mu A$
INPUT CURRENT					
Leakage, I_L (Except CLOCK)			10	μA	$0 \leq V_{IN} \leq V_{CC}$
Leakage, I_L (CLOCK Only)			50	μA	$0 \leq V_{IN} \leq V_{CC}$
INPUT CAPACITANCE					
Data		10		pF	@ 1MHz
LD/$\overline{SH}$		20		pF	@ 1MHz
CLOCK		25		pF	@ 1MHz
POWER SUPPLY CURRENT					
I_{CC}		100		mA	
A.C. CHARACTERISTICS					
See Figure 6, 7					

Maximum Guaranteed Ratings* (Valores Máximos Garantidos)

Operating Temperature Range .... 0°C to + 70°C
Storage Temperature Range .... −55°C to +150°C
Lead Temperature (soldering, 10 sec.) .... +325°C
Positive Voltage on any Pin, with respect to ground .... +8.0V
Negative Voltage on any Pin, with respect to ground .... −0.3V

* COMENTÁRIO

Esforços maiores do que os especificados podem danificar permanentemente o componente. Estes são apenas valores máximos de esforço, e a operação funcional do componente nestas condições não está prevista.

NOTA:

Quando alimentar este componente com fontes de alimentação de laboratório ou do sistema, é importante que não se ultrapassem os Valores Máximos Absolutos ou poderá ocorrer falha. Algumas fontes de alimentação apresentam perturbações nas saídas (spikes, glitches) quando são ligadas ou desligadas. Além disto, transientes de voltagem na rede CA podem aparecer na saída DC. Por exemplo, a fonte de alimentação de bancada programada para fornecer +12 volts pode ter grandes transientes de tensão quando a alimentação CA é ligada ou desligada. Se isto por acaso estiver existindo, sugere-se a utilização de um circuito de limitação (Clamp circuit).

SYMBOL	PARAMETER	CRT 8002A		CRT 8002B		CRT 8002C		UNITS
		MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
VDC	Video Dot Clock Frequency	1.0	20	1.0	15	1.0	10	MHz
PW_H	VDC—High Time	15.0		23		40		ns
PW_L	VDC—Low Time	15.0		23		40		ns
t_{CY}	LD/$\overline{SH}$ cycle time	400		533		800		ns
t_r, t_f	Rise, fall time		10		10		10	ns
t_{SET-UP}	Input set-up time	≥0		≥0		≥0		ns
t_{HOLD}	Input hold time	15		15		15		ns
t_{PD1}, t_{PD0}	Output propagation delay	15	50	15	65	15	100	ns
t_1	LD/$\overline{SH}$ set-up time	10		15		20		ns
t_2	LD/$\overline{SH}$ hold time	15		15		15		ns

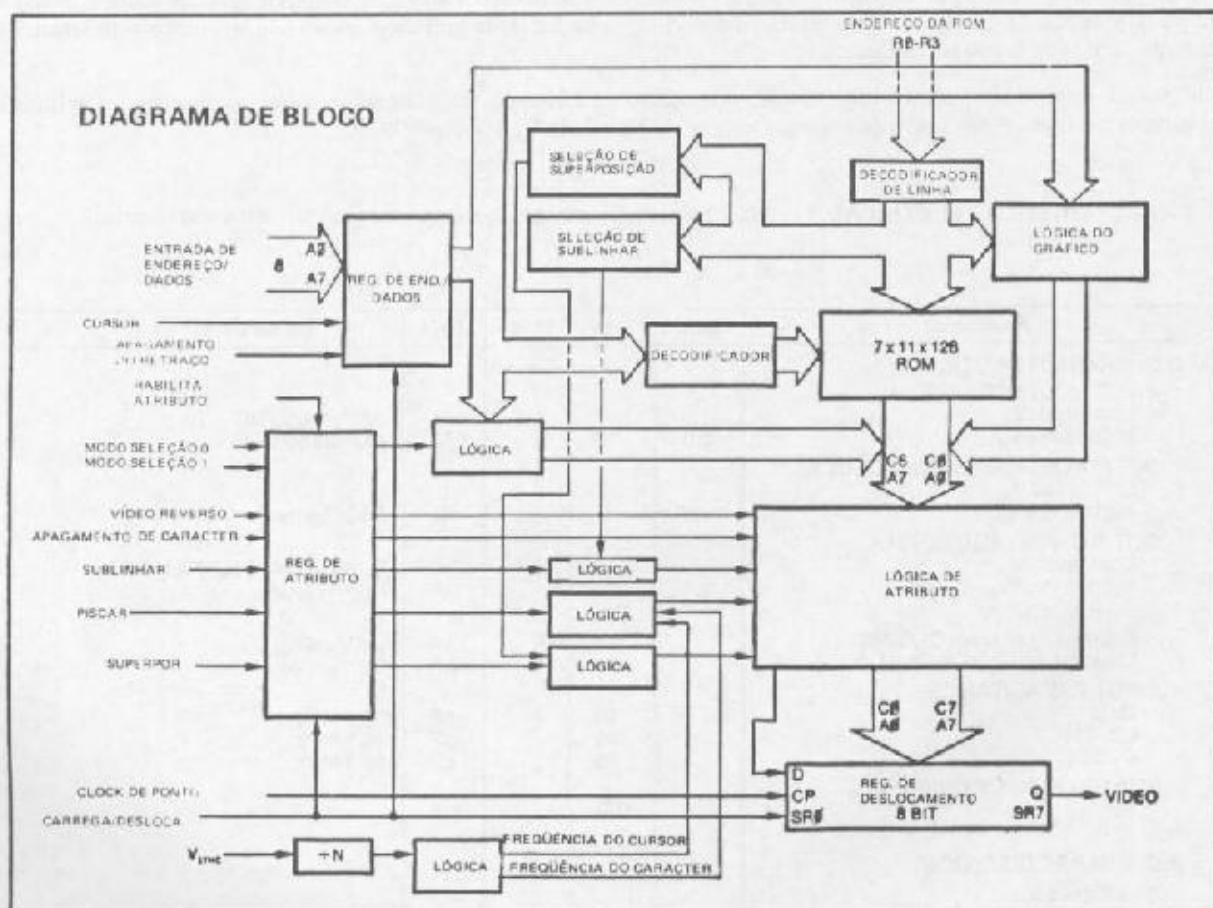
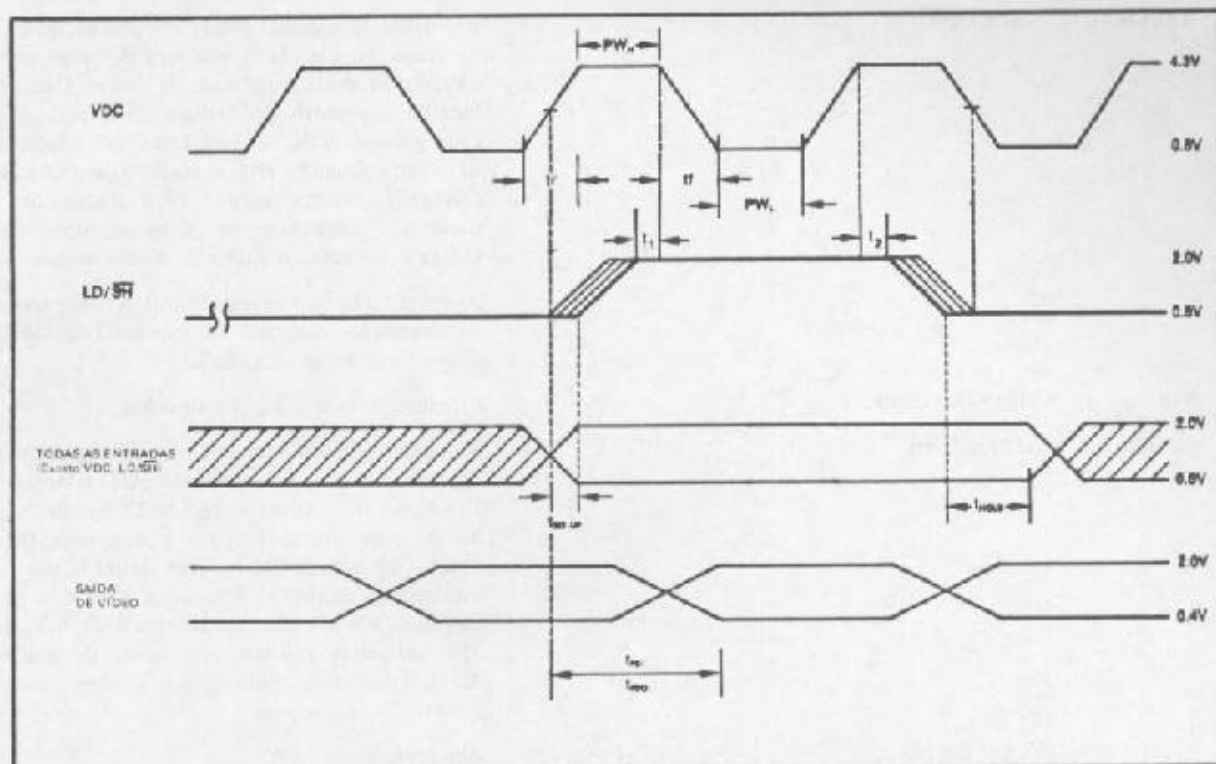



Figura 7 Diagrama de tempo CA

DESCRIÇÃO DA FUNÇÃO DOS PINOS

Nº PINO	SÍMBOLO	NOME	ENTRADA/ SAÍDA	FUNÇÃO
1	Video	Video Output	S	A saída de vídeo contém o feixe de pontos para a linha selecionada do caracter, alfanumérico, gráfico espesso/fino, ou externo, após ser processado pela lógica de atributo e pelas entradas de apagamento do retraço e cursor. No modo alfanumérico os caracteres são programados em ROM em 77 pontos (7 X 11) alocados para cada um dos 128 caracteres. (Vide figura 5.) A linha superior (R0) e as linhas de R12 a R15 são normalmente zero, assim como a coluna C7 (N.T. por causa do espaçamento entre caracteres e linhas de caracteres). Deste modo, um caracter é definido pelo retângulo limitado por R1 a R11 e C0 a C6. Quando uma linha da ROM, através da lógica de atributo, é carregada paralelamente no registro de deslocamento de 8 bits, o primeiro bit a ser serializado é o C7 (um "zero"; ou um "um" em vídeo reverso). É seguido de C6, C5 ... C0. A temporização do pulso de Load/Shift irá determinar o número de preenchimento de zeros adicionais (–, zero a N) ou "uns" do caso vídeo reverso, a serem serializados. Veja figura 4. Quando aparece o próximo pulso Load/Shift, a próxima linha de caracter da ROM, via lógica de atributo, é carregada paralelamente no registro de deslocamento e o ciclo se repete.

Pino	Sigla	Nome	Tipo	Descrição
2	LD/SH	Load/Shift	E	A entrada Load/Shift estabelece se o modo do registro de deslocamento de 8 bits será de carga paralela de entrada ou saída serializada de dados. Quando baixa, habilita o registro de deslocamento para serializar a cada pulso de VDC – Video Dot Clock (clock do ponto de vídeo). Quando alta, o registro de deslocamento é carregado (externamente) com dados de entrada paralelas, sincronamente com o próximo VDC. Durante a carga, o fluxo de dados seriais é inibido. As entradas de endereço/dado (A0–A7) são armazenadas na transição negativa do pulso Load/Shift. Veja diagrama de tempo, figura 7.
3	VDC	Video Dot Clock	E	Freqüência de serialização do vídeo.
4-11	A0–A7	Address/Data	E	No modo alfanumérico os 7 bits nas entradas (A0–A6) são decodificadas internamente para endereçar um dos 128 caracteres existentes (A7 =X). No modo externo, A0–A7 são utilizadas para inserir uma palavra de 8 bits, de uma ROM externa definida pelo usuário, na lógica de atributo da pastilha. No modo de gráfico espesso, A0–A7 são usadas para definir uma das 256 entidades gráficas. No modo de gráfico fino, A0–A2 são usadas para definir os segmentos de três linhas.
12	V_{CC}	Power Supply	F.A.	Alimentação de +5V.
13, 14, 15, 16	R2, R3, R1, R0	Row Address	E	Estas 4 entradas binárias definem o endereço de linha no bloco de caracter atual.
17	GND	Ground	GND	Terra
18	ATTBE	Attribute Enable	E	Um nível positivo nesta entrada habilita os dados das entradas de vídeo reverso, apagamento de caracter: sublinhado, superposto, piscando; modo de seleção 0 e 1 a serem armazenados no registro de atributo na transição negativa do pulso Load/Shift. A carga deste registro é inibida quando esta entrada volta para alto. Para facilitar o armazenamento do atributo de um caracter numa base de caracteres, prenda ATTBE em alto. Veja diagrama de tempo, figura 7.
19	STKRU	Strike-Thru	E	Quando esta entrada está alta RETBL = 0, as entradas paralelas do registro de deslocamento são forçadas em alto (SR0–SR7), gerando uma linha sólida no bloco de caracter. A operação de superposição (Strike-Thru) é modificada pelo vídeo reverso (veja tabela 1). Além disso, existe um decodificador programável de ROM interno, para decodificar o número da linha em que a superposição irá ser colocada, bem como para programar a altura da superposição para ser de 1 a N linhas de varredura. Atualmente, a lógica do decodificador de superposição (programável) permite ser a superposição de um número qualquer de linhas horizontais escolhidas dentro de um bloco de caracteres. A superposição padrão é uma linha dupla em cima da linha de caracter R5 e R6.
20	UNDLN	Underline	E	Quando esta entrada está alta e RTBL = 0, as entradas paralelas do registro de deslocamento são forçadas em alto (SR0–SR7), gerando uma linha sólida no bloco do caracter. A operação de sublinhar é modificada pelo vídeo reverso (veja tabela 1). Além disso, existe um decodificador programável de ROM interno, para decodificar o número da linha em que o sublinhamento será feito, bem como programar a altura do sublinhado para ser de 1 a N linhas de varredura. Atualmente, a lógica do decodificador de sublinhamento (programável) permite que o sublinhamento seja de um número qualquer de linhas horizontais escolhidas dentro de um bloco de

caracteres. O sublinhamento padrão é uma linha simples em R11.

21	REVID	Reverso Vídeo	E	Quando esta entrada está alta e RTBL =0, os dados da lógica de atributo são apresentados diretamente às entradas paralelas do registro de deslocamento. Quando o vídeo reverso está alto, os dados na lógica de atributo são invertidos e só então apresentados às entradas paralelas do registro de deslocamento (veja tabela 1).
22	CHABL	Character Blank	E	Quando esta entrada está alta, as entradas paralelas para o registro de deslocamento são colocadas em baixo, gerando uma linha de um caracter apagado. O apagamento do caracter será prioritário à piscagem. A operação de apagamento de caracter será modificada pela entrada de vídeo reverso (veja tabela 1).
23	V SYNC	V SYNC	E	Esta entrada é utilizada como entrada de clock para os dois divisores programáveis de taxa de piscagem. A taxa de piscagem do cursor (ciclo de trabalho 50/50) será o dobro da taxa de piscagem do caracter (ciclo de trabalho 75/25). Os divisores podem ser programados de ÷ 4 a ÷ 30 para o cursor e ÷ 8 a ÷ 60 para o caracter.
24	BLINK	Blink	E	Quando esta entrada está alta e RETBL = 0 e CHABL =0, o caracter irá piscar na taxa programada. A piscagem é feita e apagando-se o bloco de caracteres com o clock interno de piscar caracteres. A taxa de piscagem padrão é de 1,875 Hz.
25 26	MS1 MS0	Mode Select 1 Mode Select 0	E E	Estas 2 entradas definem 4 modos de operação do CRT 8002 como se segue:

MS1	MS0	MODO
1	1	alfanumérico
1	0	gráfico fino
0	1	modo externo
0	0	gráfico espesso

Modo alfanumérico:

Neste modo, os endereços A0–A6 (A7 = 0) são internamente decodificados para endereçar um dos 128 caracteres existentes na ROM. O caracter endereçado com a sua linha decodificada irá definir uma saída de 7 bits da ROM para ser carregada no registro de deslocamento via lógica de atributo.

Modo de gráfico fino: Neste modo, A0–A2 (A3–A7 = X) serão carregados na lógica de gráficos finos com o endereço da linha. Esta lógica definirá os segmentos de uma entidade gráfica como definido na figura 2. O topo da entidade irá começar na linha programada por máscara.

Modo externo: Neste modo, as entradas A0–A7 irão diretamente do armazenador de caracteres para o registro de deslocamento via lógica de atributo. Para tal, o usuário deverá definir fontes de caracteres externas ou entidades gráficas numa PROM, ROM ou RAM externa. Veja figura 3.

Modo de gráfico espesso: Neste modo, as entradas A0–A7 definirão uma entidade gráfica como descrito na figura 1. Cada linha da entidade gráfica será determinada pela lógica de gráfico espesso juntamente com as entradas de linha R0 a R3. Neste modo, cada segmento da entidade é definido por um dos 8 bits da palavra. Portanto, os 8 bits podem definir qualquer uma das 256 entidades gráficas possíveis. Estas entidades podem ser colocadas umas contra as outras para formar uma padronagem contígua, ou podem ser espaçados com caracteres alfanuméricos. Cada entidade ocupa o espaço de um bloco de caracter e por isto requer um byte de memória. Estes 4 modos podem ser mixados em base de caracteres.

27	CURSOR	Cursor	E	Quando esta entrada é ativada, um dos 4 pré-programados modos de cursor será ativado. O modo do cursor é programado na pastilha. O cursor padrão irá piscar numa taxa de 3,75 Hz em bloco de vídeo reverso. Os 4 modos de cursor são: **Sublinhado** – Neste modo, um sublinhado aparece na posição programada (1 a N linhas de varredura). **Sublinhado piscando** – Neste modo, o sublinhado pisca na taxa do cursor. **Bloco de vídeo reverso** – Neste modo, o bloco de caracteres é colocado em vídeo reverso. **Bloco de vídeo reverso piscando** – Neste modo, o bloco de caracteres colocado em vídeo reverso pisca na taxa do cursor. O bloco de caracteres irá alternar entre vídeo normal e vídeo reverso.
28	RETBL	Retrace Blank	E	Quando esta entrada é colocada alta, as entradas paralelas do registro de deslocamento são incondicionalmente limpas com zeros e carregadas no registro de deslocamento no próximo pulso de Load/Shift. Isto faz o vídeo apagar, independentemente de qualquer atributo, durante os retraços horizontal e vertical.

TABELA 1

CURSOR	RETBL	REVID	CHABL	UNDLN*	FUNÇÃO
X	1	X	X	X	"0" S.R. Todos
0	0	0	0	0	D (S.R.) Todos
0	0	0	0	1	"1" (S.R.)* D (S.R.) Todos os outros
0	0	0	1	X	"0" (S.R.) Todos
0	0	1	0	0	$\overline{D}$ (S.R.) Todos
0	0	1	0	1	"0" (S.R.)* $\overline{D}$ (S.R.) Todos os outros
0	0	1	1	X	"1" (S.R.) Todos
Sublinhar*	0	0	0	X	"1" (S.R.)* D (S.R.) Todos os outros
Sublinhar*	0	0	1	X	"1" (S.R.)* "0" (S.R.) Todos os outros
Sublinhar*	0	1	0	X	"0" (S.R.)* D (S.R.) Todos os outros
Sublinhar*	0	1	1	X	"0" (S.R.)* "1" (S.R.) Todos os outros
Piscar ** Sublinhar*	0	0	0	X	"1" (S.R.)* Piscando D (S.R.) Todos os outros
Piscar ** Sublinhar*	0	0	1	X	"1" (S.R.)* Piscando "0" (S.R.) Todos os outros
Piscar ** Sublinhar*	0	1	0	X	"0" (S.R.)* Piscando D (S.R.) Todos os outros
Piscar ** Sublinhar*	0	1	1	X	"0" (S.R.)* Todos os outros "1" (S.R.) Todos os outros
Bloco de VÍDEO REVERSO	0	0	0	0	$\overline{D}$ (S.R.) Todos
Bloco de VÍDEO REVERSO	0	0	0	1	"0" (S.R.)* $\overline{D}$ (S.R.) Todos os outros
Bloco de VÍDEO REVERSO	0	0	1	X	"1" (S.R.) Todos
Bloco de VÍDEO REVERSO	0	0	0	1	"0" (S.R.)* $\overline{D}$ (S.R.) Todos os outros
Bloco de VÍDEO REVERSO	0	1	0	0	D (S.R.) Todos
Bloco de VÍDEO REVERSO	0	1	0	1	"1" (S.R.)* D (S.R.) Todos os outros
Bloco de VÍDEO REVERSO	0	1	1	X	"0" (S.R.) Todos
Piscar** Bloco de VÍDEO REVERSO	0	0	0	0	Alterna vídeo normal/REVID na freqüência do cursor
Piscar** Bloco de VÍDEO REVERSO	0	0	0	1	
Piscar** Bloco de VÍDEO REVERSO	0	0	1	X	
Piscar** Bloco de VÍDEO REVERSO	0	1	0	0	
Piscar** Bloco de VÍDEO REVERSO	0	1	0	1	
Piscar** Bloco de VÍDEO REVERSO	0	1	1	X	
Piscar** Bloco de VÍDEO REVERSO					

* Na linha seleciondа pela decodificação.
** Na taxa de piscagem do cursor.

Nota: Se o caracter estiver piscando na taxa de caracter, o cursor mudará para taxa de piscagem do cursor.

FIGURA 5
FORMATO DO BLOCO DE CARACTER DA ROM

	*C7	C6	C5	C4	C3	C2	C1	CØ	—	—	LINHAS	R3	R2	R1	RØ
(TODOS ZEROS) →	0	0	0	0	0	0	0	0	—	—	RØ	0	0	0	0
77 BITS (7 x 11 ROM)	0	0	0	0	0	0	0	0	—	—	R1	0	0	0	1
	0	0	0	0	0	0	0	0	—	—	R2	0	0	1	0
	0	0	0	0	0	0	0	0	—	—	R3	0	0	1	1
	0	0	0	0	0	0	0	0	—	—	R4	0	1	0	0
	0	0	0	0	0	0	0	0	—	—	R5	0	1	0	1
	0	0	0	0	0	0	0	0	—	—	R6	0	1	1	0
	0	0	0	0	0	0	0	0	—	—	R7	0	1	1	1
	0	0	0	0	0	0	0	0	—	—	R8	1	0	0	0
	0	0	0	0	0	0	0	0	—	—	R9	1	0	0	1
	0	0	0	0	0	0	0	0	—	—	R1Ø	1	0	1	0
	0	0	0	0	0	0	0	0	—	—	R11	1	0	1	1
(TODOS ZEROS)	0	0	0	0	0	0	0	0	—	—	R12	1	1	0	0
	0	0	0	0	0	0	0	0	—	—	R13	1	1	0	1
	0	0	0	0	0	0	0	0	—	—	R14	1	1	1	0
	0	0	0	0	0	0	0	0	—	—	R15	1	1	1	1

* COLUNA 7 É TODA DE ZEROS (REVID = 0)
COLUNA 7 SERIALIZADA PRIMEIRO

EXTENSÃO DE ZEROS (BACK FILL) PARA ESPAÇAMENTO INTERCARACTER (NÚMERO CONTROLADO POR LD/$\overline{SH}$ E TEMPO DE VDC)

Consultar Fábrica para outras Fontes e Formas de Programação Opcional

FIGURA 1
MODO DE GRÁFICO ESPESSO
MSØ=Ø MS1=Ø
ENDEREÇO DA LINHA
5 BITS** N BITS**
0000
3 LINES*
3 LINES
3 LINES
3 LINES
A7 A6 A5 A4 A3 A2 A1 AØ
*ON CHIP ROM PROGRAMMABLE TO 2, 3, OR 4 LINE MULTIPLES
**CAN BE PROGRAMMED FROM 1 TO 7 BITS
***LENGTH DETERMINED BY LD/SR, VDC TIMING
EXAMPLE: 10010110
C7 C6 C5 C4 C3 C2 C1 CØ BF BF...
RØ R1 R2 R3 R4 R5 R6 R7 R8 R9 R1Ø R11 R12 R13 R14 R15
A7 A3 A6 A2 A5 A1 A4 AØ
NOTE: Unselected raster line rows are always filled with ones.
BF = back fill

FIGURA 2
MODO DE GRÁFICO FINO
MSØ=Ø MS1=1
N BITS
LINHA 0000
LINHA PROGRAMÁVEL
X X X X X A2 A1 AØ
X = DON'T CARE
* THE INSIDE SEGMENT IS MASK PROGRAMMABLE TO ROW 0000
** LENGTH DETERMINED BY LD/SR, VDC TIMING
C7 C6 C5 C4 C3 C2 C1 CØ BF BF...
RØ R1 R2 R3 R4 R5 R6 R7 R8 R9 R1Ø R11 R12 R13 R14 R15
AØ A2 A1
NOTE
NOTE: When A1 = "1", the underline row/rows are deleted. When A1 = "0", the underline, if selected, will appear.
BF = back fill

FIGURA 3
MODO EXTERNO
MSØ=1 MS1=Ø
C7 C6 C5 C4 C3 C2 C1 CØ BF BF ...
RØ – R15
A7 A6 A5 A4 A3 A2 A1 AØ A7 A7 ...
BF = back fill

FIGURA 4 SAÍDA DE VÍDEO TÍPICA
VDC
LD/$\overline{SH}$
VIDEO DATA
8 DOT FIELD
VIDEO DATA
9 DOT FIELD
NOTE: C_{xy}
x = character number
y = column number
Alphanumeric
External
BF = back fill

XTAL
74160
CONTADOR DE PONTO
TRANSBORDO
7404
VIDEO DOT CLOCK
CARACTER CLOCK
VIA DE DADOS BIDIRECIONAL
BARRA DE ENDEREÇO
LÓGICA DE PÁGINA
SELEÇÃO DO CHIP
STROBE DE DADOS
COLUNA DE CARACTER
LINHA DE CARACTER
BARRA DE ENDEREÇO DO CARACTER
VTAC
CRT 5027
H SYNC
V SYNC
C SYNC
SYNC HORIZ
SYNC VERT
COMPOSITE SYNC
APAGAMENTO
VIDEO
TO CRT MONITOR
CURSOR FLAG
CONTADOR DE VARREDURA
MICROPROCESSADOR
BARRA DE DADOS
ENDEREÇO
CONTROLE
SELETOR DE CIRCUITO DE MAPEAMENTO DE MEMÓRIA OPCIONAL (SE REQUERIDO)
ENDEREÇO
ASCII
ATRIBUTO DA BARRA DE DADOS
VIDEO DOT CLOCK
VDAC
CRT 8002
APAGAMENTO DO RETRAÇO
SAÍDA SERIAL
*2-PORT RAM 1K x 8 TO 4K x 8 BUFFER DE FORMATO DO CARACTER
DADOS DE SAÍDA
DADOS DE ENTRADA
*OR 1 PORT RAM WITH BI-DIRECT PORT
TIMING FROM DOT COUNTER OR CHARACTER CLOCK
RAM & ROM (FOR μP)
CRT 5027 VTAC
CRT 8002 VDAC
μP CONFIGURATION

FIGURA 6
V_{CC}
74SXX
500Ω
$\overline{CP}$ EXTERNAL
LOAD/$\overline{SHIFT}$ EXTERNAL
CLK
Q
D
74S74
VDC (to chip)
LD/$\overline{SH}$ (to chip)

APÊNDICE C10

COM 8046
COM 8046T

Baud Rate Generator
(Gerador da Taxa de Baud)
Divisor Programável

FACILIDADES

- Oscilador a cristal interno ou entrada de freqüência externa
- Fonte única +5V
- Escolha de 32 freqüências de saídas
- Compatibilidade direta com UART/USRT/ASTRO/USYNRT
- ROM reprogramável de tecnologia CLASP permite a geração de outras freqüências
- Compatibilidade MOS, TTL
- 1X Clock via saída fo/16
- Saída de freqüência do cristal via saída fx e fx/4
- Desabilitação de saídas via FENA

CONFIGURAÇÃO DOS PINOS

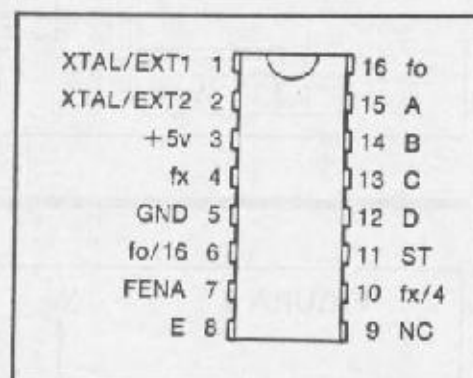
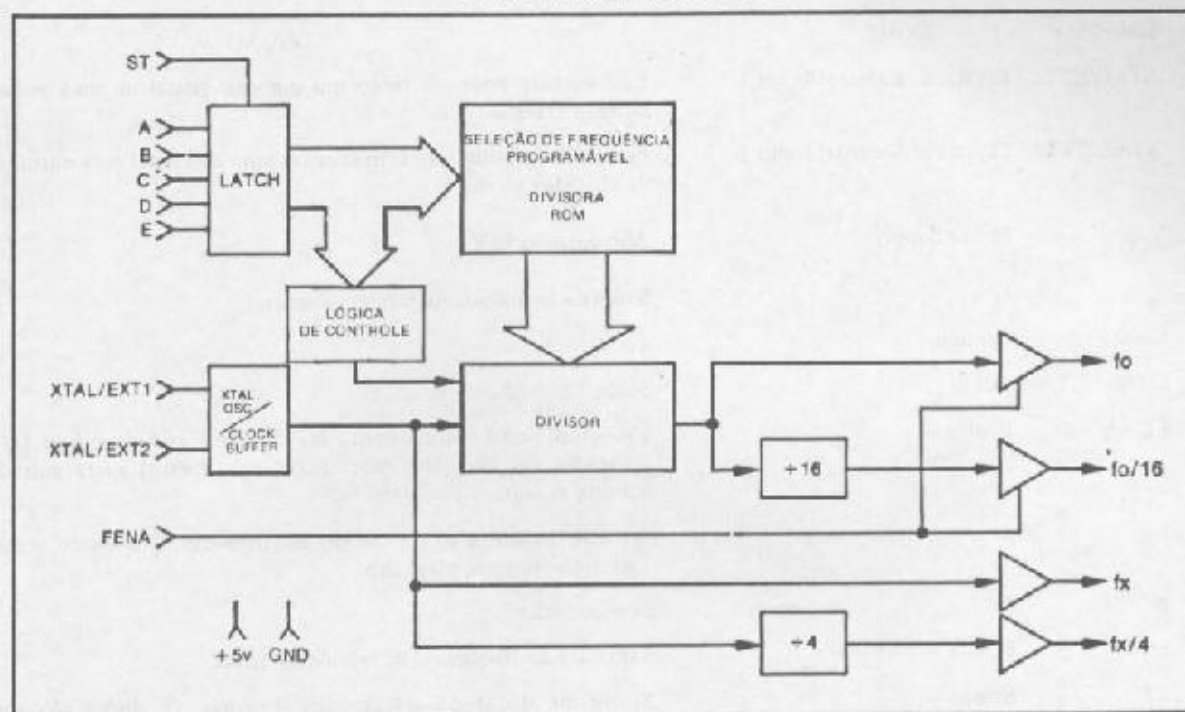



DIAGRAMA DE BLOCO

DESCRIÇÃO GERAL

O COM 8046 da SMC é uma versão aperfeiçoada do gerador de BAUD COM 5046. Ele é fabricado utilizando as tecnologias patenteadas da SMC tipo COPLAMOS e CLASP, e emprega cargas por deplexão, permitindo operação com fonte única de +5V. O COM 8046 standard é especificamente dedicado a gerar todo o espectro de 16 freqüências de comunicação de dados assíncrona/síncrona, para componentes UART/USRT/ASTRO/USYNRT de 1X, 16X e 32X.

O COM 8046 possui um oscilador a cristal interno que pode ser utilizado para fornecer a freqüência de referência principal. Como alternativa, uma freqüência externa pode ser fornecida aplicando-se sinais TTL complementares nos pinos 1 e 2. As partes utilizáveis apenas com referência externa TTL estão assinaladas COM 8046 T. As saídas TTL utilizadas para excitar o COM 8046 ou COM 8046 T não deverão ser usadas para excitar outras entradas TTL para não causar possíveis compromissos à imunidade de ruído devido à carga excessiva.

A freqüência de referência fx é usada para fornecer duas saídas de alta freqüência: uma com fx e outra com fx/4. A saída fx/4 irá excitar um 7400 padrão de carga, e a saída fx excitará 2 cargas 74LS.

A saída do oscilador/buffer é aplicada ao divisor para geração da freqüência de saída fo. O divisor é capaz de dividir por qualquer número inteiro de 6 a $2^{19} + 1$ inclusive. Se o divisor for ímpar, a saída será quadrada. Caso contrário, a saída ficará em alto pelo período de um clock fx a mais do que em baixo. A saída do divisor também é dividida internamente por 16 e fornecida no pino de saída fo. A saída fo/16 irá exercitar uma carga TTL 7400 e a saída fo duas. Ambas as saídas fo e fo/16 podem ser desabilitadas aplicando-se um nível baixo no pino FENA de entrada. Observe que a entrada FENA tem um "pull-up" interno que forçará o pino aproximadamente V_{CC} no caso de não ser conectado. A ROM divisora contém 32 divisores de 19 bits cada, e é fabricada utilizando a tecnologia CLASP exclusiva da SMC. Este processo permite a redução de "turn-around-time" para padrões de ROM.

Os 5 bits de seleção dos divisores são armazenados num latch de dados com strobe. A entrada de strobe é sensível por níveis: enquanto o strobe estiver alto, os dados passarão diretamente pela ROM. A iniciação de uma nova freqüência é efetivada com 3,5 μs de uma mudança em quaisquer dos 5 bits de seleção dos divisores; a atividade do strobe não é necessária. Esta facilidade pode ser desabilitada através de uma opção de programação CLASP, ocasionando um atraso na iniciação da nova freqüência, até o final do atual semiciclo de fo. Todas as cinco entradas de dados possuem "pull-ups" idênticos aos da entrada FENA, enquanto a entrada de strobe não.

DESCRIÇÃO DA FUNÇÃO DOS PINOS

Nº PINO	SÍMBOLO	NOME	FUNÇÃO
1	XTAL/EXT1	Crystal or External Input 1	Esta entrada pode ser tanto um pino do cristal ou uma polaridade da entrada externa.
2	XTAL/EXT2	Crystal or External Input 2	Esta entrada pode ser tanto o outro pino do cristal ou a outra polaridade da entrada externa.
3	V_{CC}	Power Supply	Alimentação +5V.
4	f_x	f_x	Saída da freqüência de referência/cristal.
5	GND	Ground	Terra
6	$f_0/16$	$f_0/16$	Saída 1X clock.
7	FENA	Enable	Um nível baixo nesta entrada faz com que as saídas f_0 e $f_0/16$ sejam colocadas em alto. Um nível baixo (ou aberto) nesta entrada FENA habilita as saídas f_0 e $f_0/16$.
8	E	E	Bit mais significativo da seleção dos divisores. Um aberto nesta entrada é equivalente a um nível alto.
9	NC	NC	Sem conexão.
10	$f_x/4$	$f_x/4$	Saída 1/4 da freqüência de referência/cristal.
11	ST	Strobe	Strobe de dados da seleção dos divisores. Os dados são amostrados quando esta entrada está alta e preservados quando esta entrada está baixa.
12-15	D, C, B, A	D, C, B, A	Bits de seleção dos divisores. A = LSB. Um aberto nestas entradas equivalem a nível alto.
16	f_0	f_0	Saída 16X clock.

CARACTERÍSTICAS ELÉTRICAS

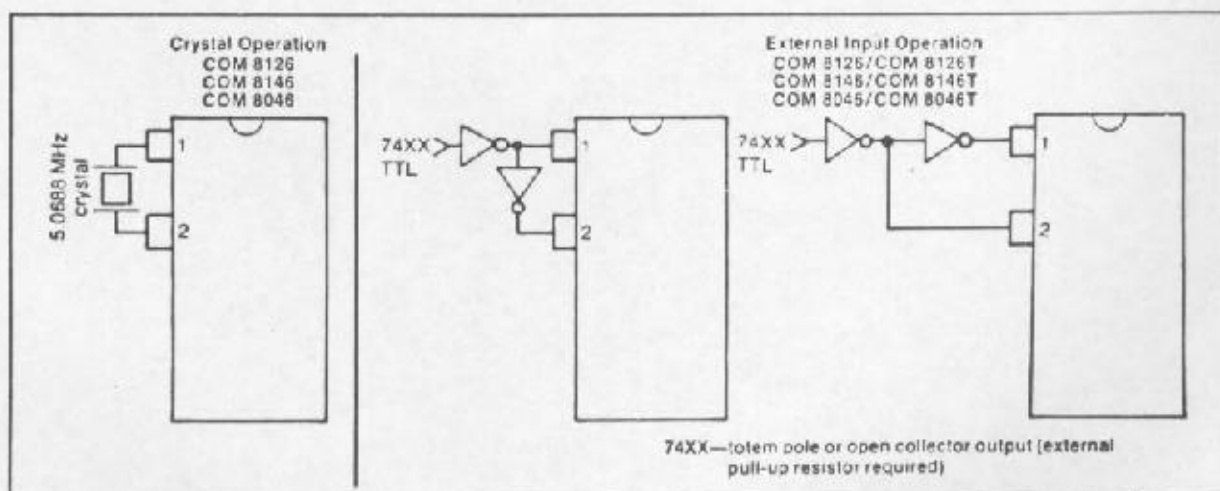
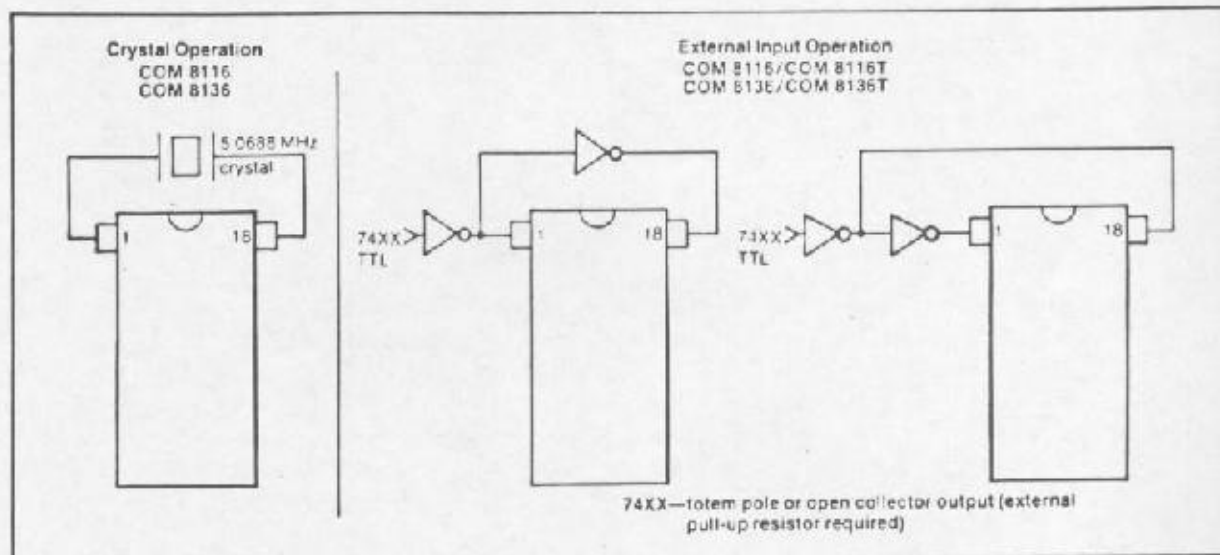
Maximum Guaranteed Ratings* (Valores Máximos Garantidos)

Operating Temperature Range .... 0°C to + 70°C
Storage Temperature Range .... −55°C to +150°C
Lead Temperature (soldering, 10 sec.) .... +325°C
Positive Voltage on any Pin, with respect to ground .... +8.0V
Negative Voltage on any Pin, with respect to ground .... −0.3V

* Esforços maiores do que os especificados podem danificar permanentemente o componente. Estes são apenas valores máximos de esforço, e a operação funcional do componente nestas condições não está prevista.

NOTA: Quando alimentar este componente com fontes de alimentação de laboratório ou do sistema, é importante que não se ultrapassem os Valores Máximos Absolutos ou poderá ocorrer falha. Algumas fontes de alimentação apresentam perturbações nas saídas (spikes, glitches) quando são ligadas ou desligadas. Além disto, transientes de voltagem na rede CA podem aparecer na saída CC. Por exemplo, a fonte de alimentação de bancada programada para fornecer + 12 volts pode ter grandes transientes de tensão quando a alimentação CA é ligada ou desligada. Se isto por acaso estiver existindo sugere-se a utilização de um circuito de limitação (Clamp Circuit).

CARACTERÍSTICAS ELÉTRICAS ($T_A = 0°C$ a $70°C$, $V_{CC} = +5V \pm 5\%$, a menos que especificado em contrário)

Parameter	Min.	Typ.	Max.	Unit	Comments
D.C. CHARACTERISTICS					
INPUT VOLTAGE LEVELS					
Low-level, V_{IL}			0.8	V	
High-level, V_{IH}	2.0			V	excluding XTAL inputs
OUTPUT VOLTAGE LEVELS					
Low-level, V_{OL}			0.4	V	$I_{OL} = 1.6mA$, for $f_X/4$, $f_O/16$
			0.4	V	$I_{OL} = 3.2mA$, for f_O, f_X, f_T
			0.4	V	$I_{OL} = 0.8mA$, for f_X
High-level, V_{OH}	3.5			V	$I_{OH} = -100\mu A$; for f_X, $I_{OH} = -50\mu A$
INPUT CURRENT					
Low-level, I_{IL}			-0.1	mA	$V_{IN} = GND$, excluding XTAL inputs
INPUT CAPACITANCE					
All inputs, C_{IN}		5	10	pF	$V_{IN} = GND$, excluding XTAL inputs
EXT INPUT LOAD		8	10		Series 7400 equivalent loads
POWER SUPPLY CURRENT					
I_{CC}			50	mA	
A.C. CHARACTERISTICS					$T_A = +25°C$
CLOCK FREQUENCY, f_{IN}	0.01		7.0	MHz	XTAL/EXT, 50% Duty Cycle ±5% COM 8046, COM 8126, COM 8146
	0.01		5.1	MHz	XTAL/EXT, 50% Duty Cycle ±5% COM 8116, COM 8136
STROBE PULSE WIDTH, t_{PW}	150		DC	ns	
INPUT SET-UP TIME					
t_{DS}	200			ns	
INPUT HOLD TIME					
t_{DH}	50			ns	
STROBE TO NEW FREQUENCY DELAY			3.5	µs	@ $f_{IN} = 5.0 MHz$

Para reprogramação da ROM, a SMC possui um programa de computador disponível onde o cliente necessita colocar apenas a freqüência de entrada e as freqüências de saída desejadas. A programação da ROM é automaticamente gerada.

Crystal Specifications

User must specify termination (pin, wire, other)
Prefer: HC-18/U or HC-25/U
Frequency — 5.0688 MHz, AT cut
Temperature range 0 C to 70 C
Series resistance 50 Ω
Series Resonant
Overall tolerance ± .01%
or as required

Crystal manufacturers (Partial List)

Northern Engineering Laboratories
357 Beloit Street
Burlington, Wisconsin 53105
(414) 763-3591

Bulova Frequency Control Products
61-20 Woodside Avenue
Woodside, New York 11377
(212) 335-6000

CTS Knights Inc.
101 East Church Street
Sandwich, Illinois 60548
(815) 786-8411

Crystek Crystals Corporation
1000 Crystal Drive
Fort Myers, Florida 33901
(813) 936-2109

COM 8046
COM 8046T

Tabela 2

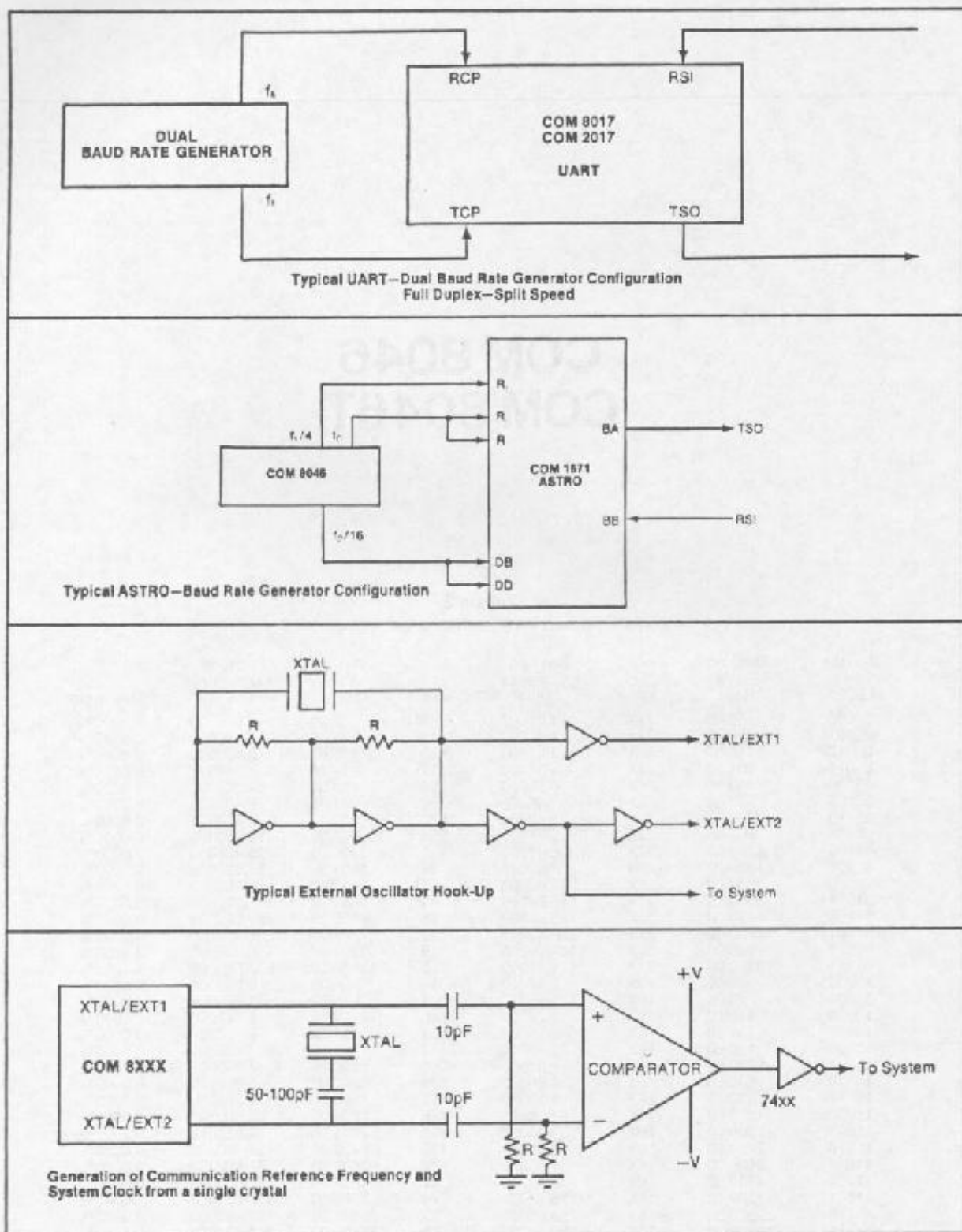
REFERENCE FREQUENCY = 5.068800MHz

Divisor Select EDCBA	Desired Baud Rate	Clock Factor	Desired Frequency (KHz)	Divisor	Actual Baud Rate	Actual Frequency (KHz)	Deviation
00000	50.00	32X	1.60000	3168	50.00	1.600000	0.0000%
00001	75.00	32X	2.40000	2112	75.00	2.400000	0.0000%
00010	110.00	32X	3.52000	1440	110.00	3.520000	0.0000%
00011	134.50	32X	4.30400	1177	134.58	4.306542	0.0591%
00100	150.00	32X	4.80000	1056	150.00	4.800000	0.0000%
00101	200.00	32X	6.40000	792	200.00	6.400000	0.0000%
00110	300.00	32X	9.60000	528	300.00	9.600000	0.0000%
00111	600.00	32X	19.20000	264	600.00	19.200000	0.0000%
01000	1200.00	32X	38.40000	132	1200.00	38.400000	0.0000%
01001	1800.00	32X	57.60000	88	1800.00	57.600000	0.0000%
01010	2400.00	32X	76.80000	66	2400.00	76.800000	0.0000%
01011	3600.00	32X	115.20000	44	3600.00	115.200000	0.0000%
01100	4800.00	32X	153.60000	33	4800.00	153.600000	0.0000%
01101	7200.00	32X	230.40000	22	7200.00	230.400000	0.0000%
01110	9600.00	32X	307.20000	16	9900.00	316.800000	3.1250%
01111	19200.00	32X	614.40000	8	19800.00	633.600000	3.1250%
10000	50.00	16X	0.80000	6336	50.00	0.800000	0.0000%
10001	75.00	16X	1.20000	4224	75.00	1.200000	0.0000%
10010	110.00	16X	1.76000	2880	110.00	1.760000	0.0000%
10011	134.50	16X	2.15200	2355	134.52	2.152357	0.0166%
10100	150.00	16X	2.40000	2112	150.00	2.400000	0.0000%
10101	300.00	16X	4.80000	1056	300.00	4.800000	0.0000%
10110	600.00	16X	9.60000	528	600.00	9.600000	0.0000%
10111	1200.00	16X	19.20000	264	1200.00	19.200000	0.0000%
11000	1800.00	16X	28.80000	176	1800.00	28.800000	0.0000%
11001	2000.00	16X	32.00000	158	2005.06	32.081013	0.2532%
11010	2400.00	16X	38.40000	132	2400.00	38.400000	0.0000%
11011	3600.00	16X	57.60000	88	3600.00	57.600000	0.0000%
11100	4800.00	16X	76.80000	66	4800.00	76.800000	0.0000%
11101	7200.00	16X	115.20000	44	7200.00	115.200000	0.0000%
11110	9600.00	16X	153.60000	33	9600.00	153.600000	0.0000%
11111	19200.00	16X	307.20000	16	19800.00	316.800000	3.1250%

Typical UART—Dual Baud Rate Generator Configuration
Full Duplex—Split Speed

Typical ASTRO—Baud Rate Generator Configuration

Typical External Oscillator Hook-Up

Generation of Communication Reference Frequency and System Clock from a single crystal

APÊNDICE D

SISTEMA OPERACIONAL DO PAZ

```
FILE  3000  7323
READY
ASSM

7324           0100 *
7324           0110 *
7324           0120 * AS IGUALDADES SEGUINTES SAO UTILIZADAS
7324           0130 * COMO CONSTANTES NO SISTEMA OPERACIONAL
7324           0140 *
7324           0150 ZERO   EQU  0
7324           0160 ONE    EQU  1
7324           0170 TWO    EQU  2
7324           0180 THREE  EQU  3
7324           0190 FOUR   EQU  4
7324           0200 FIVE   EQU  5
7324           0210 EIGHT  EQU  8D
7324           0220 ADDIS1 EQU  5              *PORTA DE ENDERECO ALTO DO DISPLAY
7324           0230 ADDIS2 EQU  6              *PORTA DE ENDERECO BAIXO DO DISPLAY
7324           0240 DATDIS EQU  7              *PORTA DE DADOS DO DISPLAY
7324           0250 EXECC  EQU  16D       *TECLA EXEC
7324           0260 NEXTC  EQU  32D       *TECLA NEXT
7324           0270 UARTIO EQU  2              *PORTA DE E/S DA UART
7324           0280 UARTST EQU  3              *PORTA DE ESTADO DA UART
7324           0290 KEYPT  EQU  0              *PORTA DE ENTRADA DO TECLADO
7324           0300 *
7324           0310 *
0000           0320        ST   0
0000           0330 *
0000           0340 *
0000           0350 *CONFIGURA O PONTEIRO DE PILHA DO SISTEMA
0000           0360 *E ENTRA COM O MODULO DE RECONHECIMENTO DE COMANDO
0000           0370 *
0000           0380 *
0000 31 C4 07  0390 COLD   LD   SP,SPSTRT  *INICIALIZA PONTEIRO DE PILHA
0003 C3 40 00  0400        JP   WARM01
0006           0410        DS   2
```

```
0008 C3 47 00     0420 WARM    JP    WARM1         *RESTART 1 (RST1) OU PARTIDA QUENTE
000B              0430         DS    5
0010 C3 C5 07     0440 RST2E   JP    RST2V         *RST 2 TRANSFERENCIA
0013              0450         DS    5
0018 C3 C8 07     0460 RST3E   JP    RST3V         *RST 3 TRANSFERENCIA
001B              0470         DS    5
0020 C3 CB 07     0480 RST4E   JP    RST4V         *RST 4 TRANSFERENCIA
0023              0490         DS    5
0028 C3 CE 07     0500 RST5E   JP    RST5V         *RST 5 TRANSFERENCIA
002B              0510         DS    5
0030 C3 D1 07     0520 RST6E   JP    RST6V         *RST 6 TRANSFERENCIA
0033              0530         DS    5
0038 C3 D4 07     0540 RST7E   JP    RST7V         *RST 7 TRANSFERENCIA
003B              0550         DS    5
0040 ED 73 DB 07  0551 WARM01 LD    (SPLSAV),SP
0044 C3 89 00     0552         JP    WARM2         *VA PARA RECONHECIMENTO DE COMANDO
0047              0560 *
0047              0570 *
0047              0580 *
0047              0590 *A PARTIDA QUENTE SALVA OS REGISTROS DE USUARIO E
0047              0600 *ENTRA O MODO DE RECONHECIMENTO DE COMANDO COM
0047              0610 *FS MOSTRADO NO DISPLAY DE DADOS E DE ENDERECOS
0047              0620 *
0047 32 E3 07     0630 WARM1   LD    (ASAV),A      *SALVA USUARIO A
004A E1           0640         POP   HL            *PEGA O PC DO USUARIO DA PILHA
004B 22 DD 07     0650         LD    (PCLSAV),HL   *SALVA O PC DO USUARIO NA AREA
004E F5           0660         PUSH  AF             DE SALVAMENTO
004F E1           0670         POP   HL            *PEGA OS "FLAGS" DO USUARIO
0050 22 E7 07     0680         LD    (ESAV),HL     *SALVA OS "FLAGS" DO USUARIO
0053 DD 22 D7 07  0690         LD    (IXLSAV),IX   *SALVA O IX DO USUARIO
0057 FD 22 D9 07  0700         LD    (IYLSAV),IY   *SALVA O IY DO USUARIO
005B ED 73 DB 07  0710         LD    (SPLSAV),SP   *SALVA O SP DO USUARIO
005F ED 57        0720         LD    A,I           *SALVA O I DO USUARIO
0061 32 DF 07     0730         LD    (ISAV),A
0064 ED 5F        0740         LD    A,R           *SALVA O R DO USUARIO
0066 32 E0 07     0750         LD    (RSAV),A
0069 21 E4 07     0760         LD    HL,BSAV
006C 70           0770         LD    (HL),B        *SALVA O B DO USUARIO
006D 23           0780         INC   HL
006E 71           0790         LD    (HL),C        *SALVA O C DO USUARIO
006F 23           0800         INC   HL
0070 72           0810         LD    (HL),D        *SALVA O D DO USUARIO
0071 23           0820         INC   HL
0072 73           0830         LD    (HL),E        *SALVA O E DO USUARIO
0073 08           0840         EX    AF,AF         *SALVA REGISTROS ALTERNADOS
0074 F5           0850         PUSH  AF             (ALT)
0075 32 EB 07     0860         LD    (AASAV),A     *SALVA ALT A
0078 22 E9 07     0870         LD    (ALSAV),HL    *SALVA ALT HL
007B E1           0880         POP   HL
007C 22 EF 07     0890         LD    (AESAV),HL    *SALVA ALT FLAGS
007F 21 EC 07     0900         LD    HL,ABSAV
0082 70           0910         LD    (HL),B        *SALVA ALT B
0083 23           0920         INC   HL
0084 71           0930         LD    (HL),C        *SALVA ALT C
0085 23           0940         INC   HL
0086 72           0950         LD    (HL),D        *SALVA ALT D
0087 23           0960         INC   HL
0088 73           0970         LD    (HL),E        *SALVA ALT E
0089              0980 *
0089              0990 *
0089              1000 *MODULO DE RECONHECIMENTO DE COMANDO
0089              1010 *
0089 CD F1 00     1020 WARM2   CALL  CLDIS         *LIMPA O DISPLAY
008C 3E FF        1030         LD    A,255D        *DISPLAY FFFF FF
008E D3 05        1040         OUT   ADDIS1
0090 D3 06        1050         OUT   ADDIS2
0092 D3 07        1060         OUT   DATDIS
0094 CD 03 01     1070         CALL  KEYIN         *PEGA CARACTER  DE ENTRADA
```

```
0097 06 40        1080         LD   B,MEM
0099 B8           1090         CP   B
009A CA F1 01     1100         JP   Z,MEMORY      *PULA SE REQUISICAO DE MEMORIA
009D 04           1110         INC  B
009E B8           1120         CP   B
009F CA 4B 02     1130         JP   Z,REGIST      *PULA SE REQUISICAO DE REGISTRO
00A2 04           1140         INC  B
00A3 B8           1150         CP   B
00A4 CA 10 03     1160         JP   Z,GOREQ
00A7 C3 89 00     1170         JP   WARM2
00AA              1180 *
00AA              1190 MEM     EQU  64D           *TECLA DE MEMORIA
00AA              1200 *
00AA              1210 *
00AA              1220 *O RESTART RESTAURA OS REGISTROS DO USUARIO
00AA              1230 *E RETORNA O CONTROLE PARA O ENDERECO
00AA              1240 *ESPECIFICADO NA POSICAO DE SALVAMENTO
00AA              1250 *DO PC NA AREA DE SALVAMENTO DE REGISTROS
00AA              1260 *
00AA 3A EC 07     1270 RESTRT LD   A,(ABSAV)     *RESTAURA TODOS OS REGISTROS
00AD 47           1280         LD   B,A
00AE 3A ED 07     1290         LD   A,(ACSAV)
00B1 4F           1300         LD   C,A
00B2 3A EE 07     1310         LD   A,(ADSAV)
00B5 57           1320         LD   D,A
00B6 3A EF 07     1330         LD   A,(AESAV)
00B9 5F           1340         LD   E,A
00BA 3A F0 07     1350         LD   A,(AFSAV)
00BD 6F           1360         LD   L,A
00BE E5           1370         PUSH HL
00BF F1           1380         POP  AF
00C0 3A EB 07     1390         LD   A,(AASAV)
00C3 2A E9 07     1400         LD   HL,(ALSAV)
00C6 D9           1410         EXX
00C7 FD 2A D9 07  1420         LD   IY,(IYLSAV) *RESTAURA IY
00CB DD 2A D7 07  1430         LD   IX,(IXLSAV) *RESTAURA IX
00CF 21 DF 07     1440         LD   HL,ISAV
00D2 7E           1450         LD   A,(HL)
00D3 ED 47        1460         LD   I,A
00D5 23           1470         INC  HL
00D6 7E           1480         LD   A,(HL)
00D7 ED 4F        1490         LD   R,A
00D9 21 E3 07     1500         LD   HL,ASAV
00DC 7E           1510         LD   A,(HL)        *RESTAURA A
00DD 23           1520         INC  HL
00DE 46           1530         LD   B,(HL)        *RESTAURA B
00DF 23           1540         INC  HL
00E0 4E           1550         LD   C,(HL)        *RESTAURA C
00E1 23           1560         INC  HL
00E2 56           1570         LD   D,(HL)        *RESTAURA D
00E3 23           1580         INC  HL
00E4 5E           1590         LD   E,(HL)        *RESTAURA E
00E5 ED 7B DB 07  1600         LD   SP,(SPLSAV) *RESTAURA PONTEIRO DA PILHA
00E9 2A DD 07     1610         LD   HL,(PCLSAV) *RECOLOCA PC NA PILHA
00EC E5           1620         PUSH HL
00ED 2A E1 07     1630         LD   HL,(LSAV)     *RESTAURA HL
00F0 C9           1640         RET                *VOLTA PARA O USUARIO
00F1              1650 *
00F1              1660 **
00F1              1670 *
00F1              1680 *"CLDIS" LIMPA OS DISPLAYS DE DADOS E
00F1              1690 *ENDERECOS, COLOCA O BUFFER DE TECLADO = 0 E
00F1              1700 *LIMPA OS FLAGS DE TECLADO
00F1              1710 *
00F1 3E 00        1720 CLDIS   LD   A,ZERO
00F3 32 F1 07     1730         LD   (KFLAGS),A   *LIMPA FLAGS
00F6 32 F2 07     1740         LD   (KDATA1),A   *LIMPA BUFFER
```

```
00F9 32 F3 07      1750          LD   (KDATA2),A
00FC D3 07         1760          OUT  DATDIS        *LIMPA DISPLAY CAMPO DE DADOS
00FE D3 05         1770          OUT  ADDIS1        *LIMPA DISPLAY CAMPO DE ENDERECOS
0100 D3 06         1780          OUT  ADDIS2
0102 C9            1790          RET
0103               1800 *
0103               1810 *
0103               1820 *"KEYIN" ESPERA POR ENTRADA DE TECLADO
0103               1830 *DETETANDO DADOS NA PORTA (0) DE ENTRADA
0103               1840 *VIA O BIT (7) DE STROBE QUE E ATIVADO PELA
0103               1850 *ENTRADA DE DADOS, O BIT DE STROBE E LIMPO, E O
0103               1860 *CARACTER  DE ENTRADA VOLTA PARA O USUARIO EM A
0103               1870 *
0103               1880 *
0103 DB 00         1890 KEYIN   IN   KEYPT         *ENTRA COM O DADO
0105 CB 7F         1900          BIT  7,A
0107 CA 03 01      1910          JP   Z,KEYIN       *FICA EM LOOP SE NAO TIVER DADO
010A 32 F4 07      1911          LD   (TEMP),A     *SALVA O CARACTER
010D DB 00         1912 KEYIN1  IN   KEYPT
010F CB 7F         1913          BIT  7,A
0111 C2 0D 01      1914          JP   NZ,KEYIN1    *PULA SE EXISTIR STROBE
0114 3A F4 07      1915          LD   A,(TEMP)
0117 CB BF         1920          RES  7,A           *LIMPA STROBE
0119 C9            1930          RET
011A               1940 *
011A               1950 *
011A               1960 *"KFLG02" ATIVA OS FLAGS DE TECLADO NEXT(0) E NO DATA(2)
011A               1970 *
011A               1980 *
011A 21 F1 07      1990 KFLG02 LD   HL,KFLAGS
011D CB C6         2000          SET  0,(HL)        *ATIVAR NEXT, NO DATA
011F CB D6         2010          SET  2,(HL)
0121 E1            2020          POP  HL            *LIMPA O RETORNO
0122 C9            2030          RET
0123               2040 *
0123               2050 *
0123               2060 *"KFLG0"  ATIVA O FLAG DE TECLADO NEXT(0)
0123               2070 *
0123               2080 *
0123 21 F1 07      2090 KFLG0   LD   HL,KFLAGS
0126 CB C6         2100          SET  0,(HL)        *ATIVAR FLAG NEXT
0128 E1            2110          POP  HL            *LIMPA O RETORNO
0129 C9            2120          RET
012A               2130 *
012A               2140 *
012A               2150 *"KFLG12"  ATIVA OS FLAGS DE TECLADO EXEC(1) E NO DATA(2)
012A               2160 *
012A               2170 *
012A 21 F1 07      2180 KFLG12 LD   HL,KFLAGS
012D CB CE         2190          SET  1,(HL)
012F CB D6         2200          SET  2,(HL)
0131 E1            2210          POP  HL            *LIMPA O RETORNO
0132 C9            2220          RET
0133               2230 *
0133               2240 *
0133               2250 *"KFLG1" ATIVA FLAG DE TECLADO EXEC(1)
0133               2260 *
0133               2270 *
0133 21 F1 07      2280 KFLG1   LD   HL,KFLAGS
0136 CB CE         2290          SET  1,(HL)        *ATIVA FLAG EXEC
0138 E1            2300          POP  HL            *LIMPA O RETORNO
0139 C9            2310          RET
013A               2320 *
013A               2330 *
013A               2340 *
013A               2350 *"ONECAR" DA ENTRADA EM UM CARACTER
013A               2360 *SEGUIDO DE UM NEXT OU EXEC DO TECLADO. VA-
013A               2370 *LIDA-O E O RETORNA PARA O USUARIO EM "KDATA2"
```

```
013A               2380 *
013A               2390 *
013A CD F1 00      2400 ONECAR CALL CLDIS       *LIMPA DISPLAY, BUFFERS E FLAGS
013D CD 03 01      2410        CALL KEYIN       *PEGO O CARACTER
0140 D3 07         2420        OUT  DATDIS      *DISPLAY O CARACTER
0142 CD 5D 01      2430        CALL CARCK1      *CHEGA O CARACTER
0145 CB 77         2440        BIT  6,A
0147 C2 51 01      2450        JP   NZ,ONECA1   *PULA SE DESLOCAMENTO
014A D6 10         2460        SUB  16D         *CARACTER = 0-F
014C F2 3A 01      2470        JP   P,ONECAR    *PULA SE NAO 0-F
014F C6 10         2480        ADD  16D
0151 32 F3 07      2490 ONECA1 LD   (KDATA2),A  *SALVA CARACTER
0154 CD 03 01      2500        CALL KEYIN       *PEGA PROXIMO CARACTER
0157 CD 6A 01      2510        CALL CARCK2
015A C3 51 01      2520        JP   ONECA1      *VA FAZER NOVAMENTE SE NAO
015D               2530 *                        EXEC OU NEXT
015D               2540 *
015D               2550 *"CARCK1" PROCURA UM NEXT OU EXEC NUM
015D               2560 *CARACTER  INICIAL SE NEXT, A ROTINA VOLTA
015D               2570 *PARA QUEM CHAMOU VIA "KFLG02". SE EXEC, A
015D               2580 *ROTINA VOLTA PARA QUEM CHAMOU VIA "KFLG12"
015D               2590 *
015D               2600 *
015D 06 20         2610 CARCK1 LD   B,NEXTC     *PROCURA UM NEXT
015F B8            2620        CP   B
0160 CA 1A 01      2630        JP   Z,KFLG02    *SE NEXT, PULA
0163 06 10         2640        LD   B,EXECC     *PROCURA UM EXEC
0165 B8            2650        CP   B
0166 CA 2A 01      2660        JP   Z,KFLG12    *SE EXEC, PULA
0169 C9            2670        RET              *SENAO RETORNA
016A               2680 *
016A               2690 *
016A               2700 *"CARCK" PROCURA POR  UM NEXT OU EXEC, ATIVA
016A               2710 *O FLAG APROPRIADO VIA KFLG0 OU KFLG1, E
016A               2720 *RETORNA AO USUARIO. SE NAO NEXT OU EXEC, A
016A               2730 *ROTINA VOLTA A ORIGEM DA CHAMADA.
016A               2740 *
016A               2750 *
016A 06 20         2760 CARCK2 LD   B,NEXTC     *PROCURA UM NEXT
016C B8            2770        CP   B
016D CA 23 01      2780        JP   Z,KFLG0     *SE NEXT, PULA
0170 06 10         2790        LD   B,EXECC     *PROCURA UM EXEC
0172 B8            2800        CP   B
0173 CA 33 01      2810        JP   Z,KFLG1
0176 C9            2820        RET
0177               2830 *
0177               2840 *
0177               2850 *"TWOCAR" DA ENTRADA A 2 CARACTERES DO
0177               2860 *TECLADO SEGUIDOS DE NEXT OU EXEC E RETORNA-OS
0177               2870 *AO USUARIO EM "KDATA2".
0177               2880 *
0177               2890 *
0177 CD A0 01      2900 TWOCAR CALL CLDAT       *LIMPA BUFFER, FLAGS E DISPLAY
017A CD 03 01      2910        CALL KEYIN       *PEGA CARACTER
017D CD 5D 01      2930        CALL CARCK1      *PROCURA NEXT OU EXEC
0180 D6 10         2940 TWOCA1 SUB  16D        *CARACTER  = 0-F
0182 F2 77 01      2950        JP   P,TWOCAR   *PULA SE NAO 0-F
0185 C6 10         2960        ADD  16D
0187 21 F3 07      2970        LD   HL,KDATA2
018A 46            2980        LD   B,(HL)      *PEGA DADO ANTIGO
018B CB 00         2990        RLC  B
018D CB 00         3000        RLC  B
018F CB 00         3010        RLC  B
0191 CB 00         3020        RLC  B
0193 80            3030        ADD  A,B         *A= ANTIGO & NOVO
0194 D3 07         3031        OUT  DATDIS     *DISPLAY A ENTRADA
0196 77            3040        LD   (HL),A      *SALVA DADO. NOVO
0197 CD 03 01      3050        CALL KEYIN       *PEGA PROXIMO CARACTER
```

```
019A CD 6A 01     3060         CALL CARCK2      *CHECA A TERMINACAO
019D C3 80 01     3070         JP   TWOCA1      *PULA SE NAO TERMINACAO
01A0              3080 *
01A0              3090 *
01A0              3100 *"CLDAT" LIMPA O BUFFER DE ENTRADA, FLAGS E DISPLAY DE DADOS
01A0              3110 *
01A0              3120 *
01A0 3E 00        3130 CLDAT  LD   A,ZERO
01A2 32 F1 07     3140        LD   (KFLAGS),A    *LIMPA FLAGS
01A5 32 F3 07     3150        LD   (KDATA2),A    *LIMPA BUFFER
01A8 32 F2 07     3160        LD   (KDATA1),A
01AB C9           3180        RET
01AC 3E 00        3181 CLADD  LD   A,ZERO       *LIMPA DISPLAY DE ENDERECOS
01AE D3 05        3182        OUT  ADDIS1
01B0 D3 06        3183        OUT  ADDIS2
01B2 C9           3184        RET
01B3              3190 *
01B3              3200 *
01B3              3210 *"FORCAR" DA ENTRADA A 4 CARACTERES
01B3              3220 *DO TECLADO SEGUIDOS DE NEXT OU EXEC E
01B3              3230 *RETORNA-OS AO USUARIO EM KDATA1 E KDATA2
01B3              3240 *
01B3              3250 *
01B3 CD A0 01     3260 FORCAR CALL CLDAT     *LIMPA FLAGS E BUFFER
01B6 CD 03 01     3270        CALL KEYIN     *PEGA CARACTER  DE ENTRADA
01B9 CD 5D 01     3280        CALL CARCK1    *PROCURA POR NEXT OU EXEC
01BC D6 10        3290 FORCA1 SUB  16D      *CARACTER   = 0-F
01BE F2 B3 01     3300        JP   P,FORCAR *PULA SE NAO 0-F
01C1 C6 10        3310        ADD  16D
01C3 32 F4 07     3320        LD   (TEMP),A      *SALVA CARACTER
01C6 3A F2 07     3330        LD   A,(KDATA1)    *A=MSD
01C9 21 F3 07     3340        LD   HL,KDATA2
01CC ED 67        3350        RRD                *AJUSTA DADO PARA NOVO CARACTER
01CE 07           3360        RLCA
01CF 07           3370        RLCA
01D0 07           3380        RLCA
01D1 07           3390        RLCA
01D2 E6 F0        3400        AND  240D          *DESLIGA DIGITO ANTIGO
01D4 21 F4 07     3410        LD   HL,TEMP
01D7 86           3420        ADD  A,(HL)        *SOMA DIGITO NOVO
01D8 2A F3 07     3430        LD   HL,(KDATA2) *SALVA LSDS NOVO
01DB 22 F2 07     3440        LD   (KDATA1),HL *SALVA MSDS NOVO
01DE 32 F3 07     3450        LD   (KDATA2),A  *SALVA LSDS NOVO
01E1 D3 06        3460        OUT  ADDIS2        *DISPLAY LSDS
01E3 3A F2 07     3470        LD   A,(KDATA1)
01E6 D3 05        3480        OUT  ADDIS1
01E8 CD 03 01     3490        CALL KEYIN         *PEGA PROXIMO CARACTER
01EB CD 6A 01     3500        CALL CARCK2        *PROCURA POR NEXT OU EXEC
01EE C3 BC 01     3510        JP   FORCA1        *PULA SE NAO NEXT OU EXEC
01F1              3520 *
01F1              3530 *
01F1              3540 *
01F1              3550 *
01F1              3560 *"MEMORY" RECEBE UM ENDERECO DO TECLADO
01F1              3570 *SEGUIDO DE DADOS COMO DEFINIDO NA SEQUENCIA
01F1              3580 *MEM (ENDERECO) NEXT, (DADO) NEXT ... (DADO) EXEC
01F1              3590 *SE O DADO E PARA SER MOSTRADO NO DISPLAY
01F1              3600 *MEM (ENDERECO) NEXT, NEXT,... NEXT, EXEC
01F1              3610 *EXEC IRA RETORNAR O CONTROLE PARA O RECONHECIMENTO
01F1              3620 *DE COMANDO
01F1              3630 *
01F1 3E 00        3640 MEMORY LD   A,ZERO       *LIMPA ENDERECO DE BASE DA MEMORIA
01F3 32 F6 07     3650        LD   (MBASE1),A
01F6 32 F7 07     3660        LD   (MBASE2),A
01F9 CD AC 01     3661        CALL CLADD
01FC CD B3 01     3670        CALL FORCAR       *PEGA ENDERECO DE BASE
01FF 3A F1 07     3680        LD   A,(KFLAGS)
0202 CB 4F        3690        BIT  1,A
```

```
0204 C2 89 00      3700            JP   NZ,WARM2     *PULA SE O FLAG EXEC ESTIVER ATIVO
0207 3A F2 07      3710            LD   A,(KDATA1)   *SALVA ENDERECO DE MEMORIA
020A 32 F7 07      3720            LD   (MBASE2),A
020D 3A F3 07      3730            LD   A,(KDATA2)
0210 32 F6 07      3740            LD   (MBASE1),A
0213 2A F6 07      3750            LD   HL,(MBASE1) *CONFIGURA ENDERECO DE BASE DE MEMORIA
0216 7E            3760 MEM1       LD   A,(HL)       *PEGA DADO DA MEMORIA
0217 D3 07         3770            OUT  DATDIS       *DISPLAY DADO DA MEMORIA
0219 CD 77 01      3780            CALL TWOCAR       *PEGA NOVO DADO
021C 3A F1 07      3790            LD   A,(KFLAGS)
021F CB 57         3800            BIT  2,A
0221 C2 43 02      3810            JP   NZ,MEM2      *PULA SE NAO DADO
0224 2A F6 07      3820            LD   HL,(MBASE1) *PEGA ENDERECO DE MEMORIA
0227 3A F3 07      3830            LD   A,(KDATA2)   *PEGA NOVO DADO
022A 77            3840            LD   (HL),A       *REPOE DADO ANTIGO
022B 3A F1 07      3850            LD   A,(KFLAGS)
022E CB 4F         3860            BIT  1,A
0230 C2 89 00      3870            JP   NZ,WARM2     *PULA SE FLAG EXEC ATIVO
0233 2A F6 07      3880 MEM12      LD   HL,(MBASE1) *INCREMENTA BASE DE MEMORIA A SOMAR
0236 23            3890            INC  HL
0237 22 F6 07      3900            LD   (MBASE1),HL
023A 7D            3901            LD   A,L
023B D3 06         3902            OUT  ADDIS2
023D 7C            3903            LD   A,H
023E D3 05         3904            OUT  ADDIS1
0240 C3 16 02      3910            JP   MEM1
0243 CB 4F         3920 MEM2       BIT  1,A
0245 C2 89 00      3930            JP   NZ,WARM2     *PULA SE FLAG EXEC ATIVO
0248 C3 33 02      3940            JP   MEM12
024B               3950 *
024B               3960 *
024B               3970 *
024B               3980 *
024B               3990 *"REGIST" DA ENTRADA A REGISTRO DO TECLADO
024B               4000 *SEGUIDO DE DADO COMO DEFINIDO NA SEQUENCIA
024B               4010 *REG(INIC.REG.)NEXT,(DADO)NEXT...(DADO)EXEC
024B               4020 *SEQUENCIA DO REGISTRO E: IX,IY,SP,PC,I,R,H,L,
024B               4030 *A,B,C,D,E,F,AL,AH,AA,AB,AC,AD,AE,AF
024B               4040 *SE APENAS O DADO E PARA SER MOSTRADO NO DISPLAY
024B               4050 *REG(INIC.REG.)NEXT, NEXT...EXEC
024B               4060 *EXEC RETORNARA O CONTROLE AO RECONHECIMENTO DE COMANDOS
024B               4070 *
024B               4080 *
024B CD 3A 01      4090 REGIST CALL ONECAR        *PEGA CARACTER  INICIAL
024E 3A F1 07      4100            LD   A,(KFLAGS)
0251 CB 57         4110            BIT  2,A
0253 C2 89 00      4120            JP   NZ,WARM2     *PULA SE FLAG NO DATA ATIVO
0256 3A F3 07      4130            LD   A,(KDATA2)    *PEGA REGISTRO BASE
0259 32 F5 07      4140 REGIO      LD   (TEMP2),A
025C CB 77         4141            BIT  6,A          *PROCURA POR DESLOCAMENTO (SHIFT)
025E C2 CC 02      4142            JP   NZ,REGISA    *PULA SE TECLA SHIFT ESTA ATIVA
0261 FE 06         4143            CP   6
0263 F2 6C 02      4144            JP   P,REGI1      *PULA SE REGISTRO DE 8 BITS
0266 3D            4145            DEC  A
0267 3D            4146            DEC  A
0268 87            4147            ADD  A            *I=(I-2)*2
0269 C3 6E 02      4148            JP   REGI2
026C 3C            4149 REGI1      INC  A
026D 3C            4150            INC  A
026E 32 F8 07      4151 REGI2      LD   (REGINX),A  *SALVA INDICE
0271 3A F5 07      4152            LD   A,(TEMP2)
0274 FE 10         4153            CP   10H
0276 FA 83 02      4154            JP   M,REGI2A
0279 CB 77         4155            BIT  6,A
027B C2 83 02      4157            JP   NZ,REGI2A     *PULA SE BIT 6 ATIVO
027E 3E 48         4158            LD   A,48H
0280 32 F5 07      4159            LD   (TEMP2),A
0283 D3 07         4160 REGI2A OUT  DATDIS        *DISPLAY SELECAO DE REGISTRO
```

```
0285 3A F8 07      4210          LD   A,(REGINX)
0288 FE 08         4220          CP   EIGHT
028A FA D6 02      4230          JP   M,XYSP        *PULA SE REGISTRO DE 16 BITS
028D 21 D7 07      4240          LD   HL,IXLSAV     *PEGA BASE A SOMAR
0290 4F            4250          LD   C,A
0291 06 00         4260          LD   B,ZERO
0293 09            4270          ADD  HL,BC

0294 22 F6 07      4280          LD   (MBASE1),HL  *SALVA REGISTRO E A SOMA
0297 7E            4290          LD   A,(HL)        *PEGA DADO DO REGISTRO
0298 D3 06         4300          OUT  ADDIS2        *DISPLAY DADO
029A 78            4310          LD   A,B
029B D3 05         4320          OUT  ADDIS1
029D CD 77 01      4330          CALL TWOCAR        *PEGA DADO NOVO
02A0 3A F1 07      4340          LD   A,(KFLAGS)
02A3 CB 57         4350          BIT  2,A
02A5 C2 B7 02      4360          JP   NZ,REGI3    *PULA SE NAO DADO
02A8 2A F6 07      4390          LD   HL,(MBASE1)
02AB 3A F2 07      4400          LD   A,(KDATA1)  *PEGA DADO NOVO
02AE 77            4410          LD   (HL),A        *REPOE DADO ANTIGO
02AF 3A F1 07      4411          LD   A,(KFLAGS)
02B2 CB 4F         4412          BIT  1,A
02B4 C2 89 00      4413          JP   NZ,WARM2      *PULA SE FLAG EXEC ATIVO
02B7 3A F5 07      4420 REGI3    LD   A,(TEMP2)     *INCREMENTA INDICE
02BA 3C            4421          INC  A
02BB 32 F5 07      4422          LD   (TEMP2),A
02BE 3A F8 07      4423          LD   A,(REGINX)    *INCREMENTA INDICE
02C1 3C            4430          INC  A
02C2 FE 1A         4440          CP   1AH
02C4 FA 6E 02      4450          JP   M,REGI2        *PULA SE INDICE MENOR QUE 1A
02C7 3E 02         4460 REGI4    LD   A,TWO         *CONFIGURA INDICE INICIAL
02C9 C3 59 02      4470          JP   REGIO
02CC D6 48         4480 REGISA   SUB  48H
02CE FA 4B 02      4490          JP   M,REGIST      *PULA SE REGISTRO INVALIDO
02D1 C6 12         4500          ADD  12H
02D3 C3 6E 02      4510          JP   REGI2
02D6 21 D7 07      4520 XYSP     LD   HL,IXLSAV
02D9 4F            4530          LD   C,A
02DA 06 00         4540          LD   B,ZERO
02DC 09            4550          ADD  HL,BC          *HL=REG SALVA ENDERECO

02DD 22 F6 07      4560          LD   (MBASE1),HL
02E0 7E            4570          LD   A,(HL)         *DISPLAY DADO DO REBISTRO
02E1 D3 06         4580          OUT  ADDIS2
02E3 23            4590          INC  HL
02E4 7E            4600          LD   A,(HL)
02E5 D3 05         4610          OUT  ADDIS1
02E7 3A F8 07      4620          LD   A,(REGINX)
02EA 3C            4630          INC  A
02EB 32 F8 07      4640          LD   (REGINX),A
02EE CD B3 01      4650          CALL FORCAR         *PEGA DADO NOVO
02F1 3A F1 07      4660          LD   A,(KFLAGS)
02F4 CB 57         4670          BIT  2,A
02F6 C2 08 03      4680          JP   NZ,REGI5    *PULA SE NAO DADO
02F9 2A F6 07      4710          LD   HL,(MBASE1) *REPOE DADO ANTIGO
02FC 3A F3 07      4720          LD   A,(KDATA2)
02FF 77            4730          LD   (HL),A
0300 3A F2 07      4740          LD   A,(KDATA1)
0303 23            4750          INC  HL
0304 77            4760          LD   (HL),A
0305 3A F1 07      4761          LD   A,(KFLAGS)
0308 CB 4F         4762 REGI5    BIT  1,A
030A C2 89 00      4763          JP   NZ,WARM2     *PULA SE FLAG EXEC ATIVO
030D C3 B7 02      4770          JP   REGI3
0310               4780 *
0310               4790 *
0310               4800 *
0310               4810 *
```

```
0310                4820 *"GO REQ" LIMPA O ENDERECO DE RESTART
0310                4830 *DO USUARIO NA AREA DE SALVAMENTO DE REGISTRO
0310                4840 *E SAI PARA O MODULO DE RESTART
0310                4850 *
0310                4860 *
0310 CD AC 01       4870 GOREQ  CALL CLADD
0313 CD B3 01       4871        CALL FORCAR       *PEGA ENDERECO DE RESTART
0316 3A F1 07       4880        LD   A,(KFLAGS)
0319 CB 57          4890        BIT  2,A
031B C2 89 00       4900        JP   NZ,WARM2     *SAI SE NAO DADO
031E 3A F3 07       4910        LD   A,(KDATA2)   *SALVA NOVO ENDERECO
0321 32 DD 07       4920        LD   (PCLSAV),A
0324 3A F2 07       4930        LD   A,(KDATA1)
0327 32 DE 07       4940        LD   (PCHSAV),A
032A C3 AA 00       4950        JP   RESTRT
032D                4960 *
032D                4970 *
032D                4980 *
032D                4990 *"UATST" E UM LOOP DA UART PARA CHECAGEM
032D                5000 *ELE UTILIZA UM LOOP COM A PORTA DE
032D                5010 *SAIDA INTERLIGADA COM A DE ENTRADA
032D                5020 *SE UM ERRO E DETECTADO, O ERRO E MOSTRADO
032D                5030 *NO DISPLAY DE ENDERECO E
032D                5040 *O CARACTER E MOSTRADO NO DISPLAY DE DADOS
032D                5050 *O CARACTER DE SAIDA E MOSTRADO NO MSD
032D                5060 *DO DISPLAY DE ENDERECO.
032D                5070 *
032D 06 00          5080 UATST  LD   B,ZERO       *
032F DB 03          5090        IN   UARTST       *PEGA ESTADO
0331 CB 47          5100        BIT  0,A
0333 CA 53 03       5110        JP   Z,UAER1      *PULA SE BUFFER DE TRANSM. NAO VAZIO
0336 78             5120 UATSTO LD   A,B          *PEGA CARACTER DE SAIDA
0337 D3 05          5130        OUT  ADDIS1
0339 D3 02          5140        OUT  UARTIO
033B DB 03          5150 UATST1 IN   UARTST
033D CB 4F          5160        BIT  1,A
033F CA 3B 03       5170        JP   Z,UATST1     *PULA SE NAO DADO DISPONIVEL
0342 E6 1C          5180        AND  1CH
0344 C2 53 03       5190        JP   NZ,UAER1     *PULA SE ERRO DE PARIDADE
0347 DB 02          5240        IN   UARTIO       *PEGA CARACTER DE ENTRADA
0349 D3 07          5250        OUT  DATDIS
034B B8             5260        CP   B
034C C2 5A 03       5270        JP   NZ,UAER2     *PULA SE ENTRADA # SAIDA
034F 04             5280        INC  B
0350 C3 36 03       5290        JP   UATSTO
0353 D3 06          5300 UAER1  OUT  ADDIS2       *DISPLAY ESTADO DA UART
0355 DB 02          5310        IN   UARTIO       *PEGA DADO DE ENTRADA
0357 D3 07          5320        OUT  DATDIS
0359 76             5330        HALT
035A 3E 0F          5340 UAER2  LD   A,0FH
035C ED 79          5350        OUT  (ADDIS2),A
035E 76             5360        HALT
035F                5370 *
035F                5380 *
035F                5390 *EXCITADOR DE ENTRADA DE TTY
035F                5400 *DA ENTRADA AO DADO NO BUFFER ESPECIFICADO
035F                5410 *A ENTRADA E FINALIZADA QUANDO UM "RETORNO
035F                5420 *DE CARRO" E DETETADO OU O NUMERO DE CARACTERES
035F                5430 *ESPECIFICADO JA FOI TRANSMITIDO PELO DISPOSITIVO.
035F                5440 *
035F 2A F9 07       5450 TTYINP LD   HL,(TTYIBF)  *PEGA ENDERECO DO BUFFER
0362 3A FD 07       5460        LD   A,(TTYIC)    *PEGA NUMERO DE CARACTER1
0365 47             5470        LD   B,A
0366 DB 03          5480 TTYIN1 IN   UARTST       *PEGA ESTADO DA UART
0368 CB 4F          5490        BIT  1,A
036A CA 66 03       5500        JP   Z,TTYIN1     *PULA SE NAO  DADO
036D E6 1C          5510        AND  1CH
036F C2 9B 03       5520        JP   NZ,TTYERR    *PULA SE ERRO DE PARIDADE
```

```
0372 DB 02          5570        IN    UARTIO        *PEGA CARACTER DE ENTRADA
0374 77             5580        LD    (HL),A        *SALVA CARACTER NO BUFFER DO USUARIO
0375 FE 0D          5590        CP    A,0DH
0376 CA 91 03       5600        JP    Z,TTYIN2      *PULA SE RETORNO DE CARRO
0379 3E 01          5610        LD    A,ONE          *ATIVA CONTAGEM DE CARACTERES DE SAIDA
037B 22 FB 07       5620 TTYIN3 LD    (TTYOBF),HL   *ATIVA ENDERECO DO BUFFER DE SAIDA
037E 32 FE 07       5630        LD    (TTYOC),A
0381 78             5631        LD    A,B
0382 32 F4 07       5632        LD    (TEMP),A
0385 CD 9E 03       5640        CALL  TTYOUT        *VAI DAR SAIDA AO CARACTER
0388 3A F4 07       5641        LD    A,(TEMP)
038B 47             5642        LD    B,A
038C 05             5650        DEC   B
038D C8             5660        RET   Z       *RETORNA SE TODOS OS CARACTERES JA ENTRADOS
038E C3 66 03       5670        JP    TTYIN1
0391 21 9C 03       5680 TTYIN2 LD    HL,LF         *PEGA ENDERECO DE PULAR DE LINHA
0394 3E 02          5690        LD    A,TWO
0396 06 01          5700        LD    B,ONE
0398 C3 7B 03       5710        JP    TTYIN3
039B C9             5720 TTYERR RET                  *RETORNA COM CODIGO DE ERRO COM
039C 0D 0A          5730 LF     DB    0DH,0AH       *RETORNO DE CARRO/SALTO DE LINHA

039E                5740 *
039E                5750 *EXCITADOR DE SAIDA DE TTY
039E                5760 *"TTYOUT" DA SAIDA AOS DADOS DO BUFFER DO
039E                5770 *USUARIO ESPECIFICADO PARA A UART O NUMERO DE
039E                5780 *CARACTERES ESPECIFICADOS E TRANSMITIDO E O CONTRO-
039E                5790 *LE VOLTA AO USUARIO
039E                5800 *
039E 2A FB 07       5810 TTYOUT LD    HL,(TTYOBF)   *PEGA ENDERECO DO BUFFER
03A1 3A FE 07       5820        LD    A,(TTYOC)     *PEGA NUMERO DE CARACTERES
03A4 47             5830        LD    B,A
03A5 0E 00          5840 TTYOU1 LD    C,ZERO
03A7 11 00 00       5850        LD    DE,ZERO
03AA DB 03          5860 TTYO1  IN    UARTST        *PEGA ESTADO
03AC CB 47          5870        BIT   0,A
03AE CA BC 03       5880        JP    Z,TTYOU2      *PULA SE BUFFER NAO VAZIO
03B1 7E             5890        LD    A,(HL)        *PEGA CARACTER
03B2 D3 02          5900        OUT   UARTIO        *DA SAIDA AO CARACTER
03B4 05             5910        DEC   B
03B5 3E 00          5920        LD    A,ZERO
03B7 C8             5930        RET   Z             *RETORNA SE BUFFER VAZIO
03B8 23             5931        INC   HL
03B9 C3 A5 03       5940        JP    TTYOU1
03BC 13             5950 TTYOU2 INC   DE            *ATRASO PARA NOVA TENTATIVA
03BD 7B             5960        LD    A,E
03BE FE 00          5970        CP    ZERO
03C0 C2 BC 03       5980        JP    NZ,TTYOU2
03C3 7A             5990        LD    A,D
03C4 FE 00          6000        CP    ZERO
03C6 C2 BC 03       6010        JP    NZ,TTYOU2
03C9 0C             6020        INC   C
03CA FE 05          6030        CP    FIVE
03CC C2 AA 03       6040        JP    NZ,TTYO1      *PULA SE MENOR QUE 5 TENTATIVAS
03CF 3E 01          6050        LD    A,ONE     *SE  NAO RETORNA COM A=1
03D1 C9             6060        RET
03D2                6070 *
07C4                6080        ST    7C4H
07C4                6090 *
07C4                6100 *PAGINA 2: CONSTANTES, AREAS DE DESVIO, E
07C4                6110 *AREA DE SALVAMENTO DE REGISTRO
07C4                6120 *
07C4                6130 SPSTRT DB    0         *AREA DA PILHA
 00
07C5                6140 *
07C5                6150 *AREA DE RESTART DO USUARIO
07C5                6160 *
07C5                6170 RST2V  DS    3         *AREA VAZIA DE USUARIO P/ RST2
```

```
07C8            6180 RST3V   DS    3        *AREA VAZIA DE USUARIO P/ RST3
07CB            6190 RST4V   DS    3        *AREA VAZIA DE USUARIO P/ RST4
07CE            6200 RST5V   DS    3        *AREA VAZIA DE USUARIO P/ RST5
07D1            6210 RST6V   DS    3        *AREA VAZIA DE USUARIO P/ RST6
07D4            6220 RST7V   DS    3        *AREA VAZIA DE USUARIO P/ RST7
07D7            6230 *
07D7            6240 *AREA DE SALVAMENTO DE REGISTROS
07D7            6250 *
07D7            6260 IXLSAV DB    0
 00
07D8            6270 IXHSAV DB    0
 00
07D9            6280 IYLSAV DB    0
 00
07DA            6290 IYHSAV DB    0
 00
07DB            6300 SPLSAV DB    0
 00
07DC            6310 SPHSAV DB    0
 00
07DD            6320 PCLSAV DB    0
 00
07DE            6330 PCHSAV DB    0
 00
07DF            6340 ISAV    DB    0
 00
07E0            6350 RSAV    DB    0
 00
07E1            6360 LSAV    DB    0
 00
07E2            6370 HSAV    DB    0
 00
07E3            6380 ASAV    DB    0
 00
07E4            6390 BSAV    DB    0
 00
07E5            6400 CSAV    DB    0
 00
07E6            6410 DSAV    DB    0
 00
07E7            6420 ESAV    DB    0
 00
07E8            6430 FSAV    DB    0
 00
07E9            6440 ALSAV   DB    0
 00
07EA            6450 AHSAV   DB    0
 00
07EB            6460 AASAV   DB    0
 00
07EC            6470 ABSAV   DB    0
 00
07ED            6480 ACSAV   DB    0
 00
07EE            6490 ADSAV   DB    0
 00
07EF            6500 AESAV   DB    0
 00
07F0            6510 AFSAV   DB    0
 00
07F1            6520 *
07F1            6530 *AREA DE ARMAZENAMENTO DE DADOS
07F1            6540 *
07F1            6550 KFLAGS DB    0         *FLAGS DO TECLADO
 00
07F2            6560 KDATA1 DB    0         *BUFFER DE ENTRADA DO TECLADO
 00
07F3            6570 KDATA2 DB    0
```

```
  00
07F4              6580 TEMP   DB   0
  00
07F5              6581 TEMP2  DB   0
  00
07F6              6590 MBASE1 DB   0     *ENDERECO DA MEMORIA DE BASE
  00
07F7              6600 MBASE2 DB   0
  00
07F8              6610 REGINX DB   0     *REGISTRO INDICE
  00
07F9              6620 TTYIBF DS   2     *ENDERECO DO BUFFER DE ENTRADA DA TTY
07FB              6630 TTYOBF DS   2     *ENDERECO DO BUFFER DE SAIDA DA TTY
07FD              6640 TTYIC  DB   0     *CONTAGEM DOS CARACTERES DE SAIDA DA TTY
  00
07FE              6650 TTYOC  DB   0     *
  00
07FF              6660 *
07FF              6670 END


FILE  3000  7323
READY
```

APÊNDICE E

ESPECIFICAÇÕES TÉCNICAS DA CPU Z80

APÊNDICE E1

ESPECIFICAÇÕES ELÉTRICAS

Absolute Maximum Ratings* (Valores Máximos Absolutos)

Temperature Under Bias	Specified operating range.
Storage Temperature	-65°C to +150°C
Voltage On Any Pin with Respect to Ground	-0.3V to +7V
Power Dissipation	1.5W

* Comentário
Esforços maiores do que os especificados nos Valores Máximos Absolutos (Absolute Max. Ratings) podem danificar permanentemente o componente. Estes são apenas valores máximos de esforço, e a operação funcional do componente nestas condições não está prevista. A exposição aos valores máximos absolutos por longos períodos afetam a confiabilidade do componente.

Note: For Z80 CPU all AC and DC characteristics remain the same for the military grade parts except I_{cc}.

I_{cc} = 200 mA

Características CC do Z80

T_A = 0°C a 70°C, V_{cc} = 5V ± 5% a menos que especificado em contrário.

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} -.6		V_{cc}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250μA
I_{CC}	Power Supply Current			150	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{cc}$

Capacitância

T_A = 25°C, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max.	Unit
C_{Φ}	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

Z80A–CPU

Informações para pedidos

C — Cerâmico
P — Plástico
S — Standard 5V ±5% a 70°C
E — Estendido 5V ±5% – 40° a 85°C
M — Militar 5V ±10% – 55° a 125°C

Características CC do Z80A

T_A = 0°C a 70°C, V_{cc} = 5V ± 5% a menos que especificado em contrário.

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} -.6		V_{cc}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250μA
I_{CC}	Power Supply Current		90	200	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 < V_{IN} < V_{cc}$

Capacitância

T_A = 25°C, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max.	Unit
C_ϕ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

Z80A-CPU

Informações para Pedidos

C — Cerâmico
P — Plástico
S — Standard 5V ±5% 0° a 70°C

Característica CA

Z80-CPU

$T_A = 0°C$ a $70°C$, $V_{CC} = +5V \pm 5\%$, a menos que especificado em contrário.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	.4	[12]	μsec	
	$t_w(\Phi H)$	Clock Pulse Width, Clock High	180	[E]	nsec	
	$t_w(\Phi L)$	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		145	nsec	$C_L = 50pF$
	$t_{F(AD)}$	Delay to Float		110	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		230	nsec	$C_L = 50pF$
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	60		nsec	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]			
	t_H	Any Hold Time for Setup Time	0		nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		100	nsec	$C_L = 50pF$
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		90	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		110	nsec	
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		110	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		100	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		130	nsec	
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		100	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		110	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		80	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		90	nsec	
	$t_{DH\Phi(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		100	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		130	nsec	$C_L = 50pF$
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		130	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		180	nsec	$C_L = 50pF$
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		150	nsec	
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	$C_L = 50pF$
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NMI}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{NMI}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		120	nsec	$C_L = 50pF$
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		110	nsec	
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	90		nsec	
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		100	nsec	
	t_{mr}	$\overline{M1}$ Stable Prior to $\overline{IORQ}$ (Interrupt Ack.)	[11]		nsec	

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$

[2] $t_{aci} = t_c - 80$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 40$

[4] $t_{caf} = t_{w(\Phi L)} + t_r - 60$

[5] $t_{dcm} = t_c - 210$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 210$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 80$

[8] $t_{w(MRL)} = t_c - 40$

[9] $t_{w(MRH)} = t_{w(\Phi H)} + t_f - 30$

[10] $t_{w(\overline{WRL})} = t_c - 40$

[11] $t_{mr} = 2t_c + t_{w(\Phi H)} + t_f - 80$

NOTAS:

A. Os dados devem ser habilitados para a barra de dados da CPU quando $\overline{RD}$ está ativo. Durante o reconhecimento de interrupção os dados serão habilitados quando $\overline{M1}$ e $\overline{IORQ}$ estão ativos.

B. Todos os sinais de controle são sincronizados internamente, então eles podem estar totalmente assíncronos com relação ao CLOCK.

C. O sinal $\overline{RESET}$ deve ficar ativo pelo menos por três períodos de clock.

D. Atraso de saída X capacitância de carga
$T_A = 70°C$ e $V_{CC} = +5V \pm 5\%$
somar 10 nseg. de atraso para cada 50pf adicionados até um máximo de 200pf para a barra de dadados, e 100pf para linha de endereço/controle.

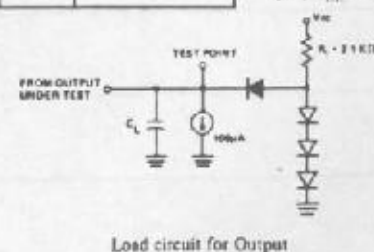
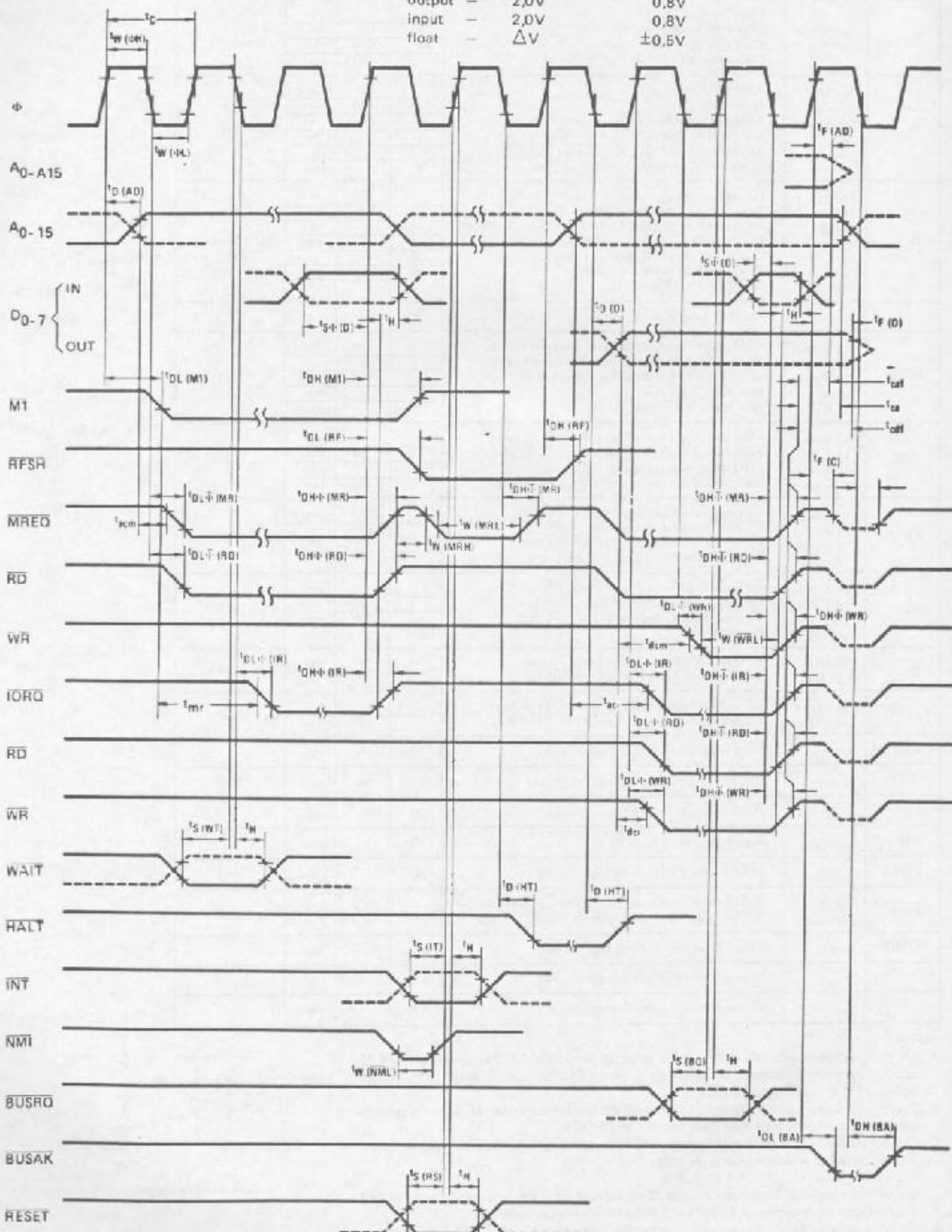
E. Embora seja estático por projeto, os testes garantem t_w de 200μ seg. máximo.

Load circuit for Output

DIAGRAMA DE TEMPO CA:

As medidas de tempo são feitas nas seguintes tensões,
a menos que especificado em contrário:

		"1"	"0"
clock	–	V_{CC} – 0,6V	0,45V
output	–	2,0V	0,8V
input	–	2,0V	0,8V
float	–	ΔV	±0,5V

Características CA

Z80A-CPU

$T_A = 0°C$ a $70°C$, $V_{cc} = +5V \pm 5\%$, a menos que especificado em contrário.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	.25	[12]	μsec	
	$t_w(\Phi H)$	Clock Pulse Width, Clock High	110	[E]	nsec	
	$t_w(\Phi L)$	Clock Pulse Width, Clock Low	110	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		110	nsec	$C_L = 50pF$
	$t_{F(AD)}$	Delay to Float		90	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		150	nsec	$C_L = 50pF$
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	35		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	50		nsec	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]			
	t_H	Any Hold Time for Setup Time		0	nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		85	nsec	$C_L = 50pF$
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		85	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		85	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		75	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		85	nsec	
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		85	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		85	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		85	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		95	nsec	
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		85	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		85	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		65	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		80	nsec	
	$t_{DH\Phi(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		80	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		100	nsec	$C_L = 50pF$
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		100	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		130	nsec	$C_L = 50pF$
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		120	nsec	
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	$C_L = 50pF$
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NMI}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{NMI}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	50		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		100	nsec	$C_L = 50pF$
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		100	nsec	
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	60		nsec	
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		80	nsec	
	t_{mr}	$\overline{M1}$ Stable Prior to $\overline{IORQ}$ (Interrupt Ack.)	[11]		nsec	

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 65$

[2] $t_{aci} = t_c - 70$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 50$

[4] $t_{caf} = t_{w(\Phi L)} + t_r - 45$

[5] $t_{dcm} = t_c - 170$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 170$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 70$

[8] $t_{w(\overline{MRL})} = t_c - 30$

[9] $t_{w(\overline{MRH})} = t_{w(\Phi H)} + t_f - 20$

[10] $t_{w(\overline{WRL})} = t_c - 30$

[11] $t_{mr} = 2t_c + t_{w(\Phi H)} + t_f - 65$

NOTAS:

A. Os dados devem ser habilitados para a barra de dados da CPU quando $\overline{RD}$ está ativo. Durante o reconhecimento de interrupção os dados serão habilitados quando $\overline{M1}$ e $\overline{IORQ}$ estão ativos.

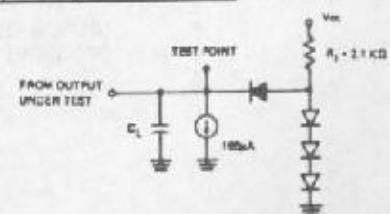
B. Todos os sinais de controle são sincronizados internamente, então eles podem estar totalmente assíncronos com relação ao CLOCK.

C. O sinal $\overline{RESET}$ deve ficar ativo pelo menos por três períodos de clock.

D. Atraso de saída X capacitância de carga

$TA = 70°C$ e $Vcc = +5V \pm 5\%$

Somar 10nseg. de atraso para cada 50pf adicionados até um máximo de 200pf para a barra de dados, e 100pf para linha de endereço/controle.

E. Embora seja estático por projeto, os testes garantem $t_w(\Phi H)$ de 200 μ seg. máximo.

Load circuit for Output

APÊNDICE E2

TEMPORIZAÇÃO DE CPU

A CPU Z80 executa instruções passando por um conjunto muito preciso de operações básicas. Estas incluem:

Escritura ou leitura de memória
Escrita ou leitura de dispositivo de E/S
Reconhecimento de interrupção

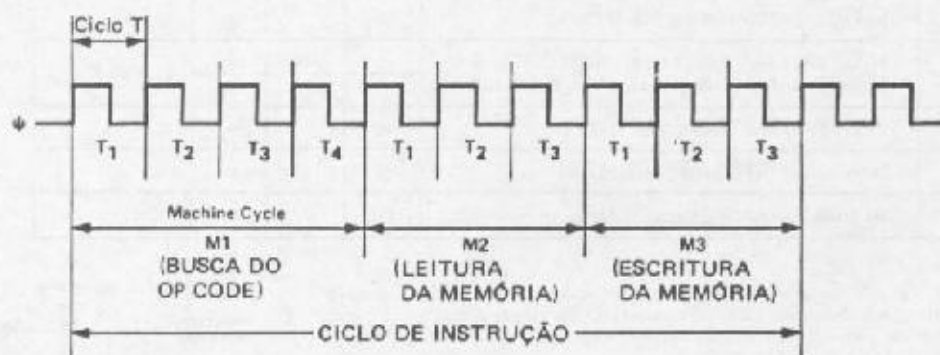
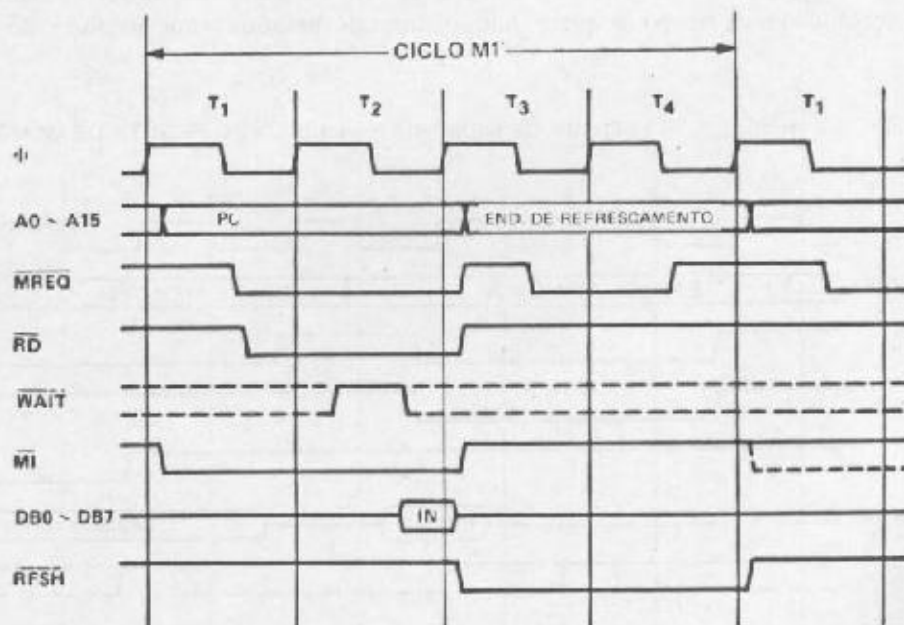
Todas as instruções são meramente uma série destas operações básicas. Cada uma destas operações básicas pode levar de 3 a 6 períodos de clock para completar-se ou elas podem ser estendidas para sincronizar a CPU com a velocidade de dispositivos externos. Os períodos básicos de clock são referidos como ciclos "T", e as operações básicas são referidas como ciclos M (de máquina). A figura 0 ilustra como uma instrução típica pode ser meramente uma série de ciclos M e T. Observe que esta instrução consiste de 3 ciclos de máquina (M1, M2 e M3). O primeiro ciclo de máquina de qualquer instrução é um ciclo de busca (fetch) que tem 4, 5 ou 6 ciclos T de duração (a menos que seja estendida pelo sinal de WAIT que será descrito em detalhes na próxima seção). O ciclo de busca (M1) é usado para buscar o código de operação (OP CODE) da próxima instrução a ser executada. Os ciclos de máquina subseqüentes movem dados entre a CPU e a memória ou dispositivos de E/S, e eles podem ter de 3 a 5 ciclos T (novamente eles podem ser extendidos por estados de WAIT). Os parágrafos seguintes descrevem a temporização que ocorre em cada um dos ciclos de máquina básicos.

Exemplo de Tempos Básicos de CPU

Figura 0

Toda a temporização da CPU pode ser dividida em alguns poucos diagramas de tempo simples como mostrado nas figuras de 1 a 7. Estes diagramas mostram as seguintes operações básicas com e sem estados de WAIT (espera). Os estados de WAIT são adicionados para sincronizar a CPU com memórias lentas ou dispositivos de E/S.

1. Busca de OP CODE de instrução (ciclo M1)
2. Ciclos de escrita ou leitura de dados na memória
3. Ciclos de escrita ou leitura de E/S
4. Ciclos de pedidos/reconhecimento de barra
5. Ciclo de pedido/reconhecimento de interrupção
6. Ciclo de pedido/reconhecimento de interrupção não mascarável
7. Saída de uma instrução de HALT

BUSCA DE INSTRUÇÃO

A figura 1 mostra os tempos durante um ciclo M1 (busca do OP CODE). Observe que o PC é colocado na barra de endereços no início do ciclo M1. Meio tempo de clock e depois o sinal $\overline{MREQ}$ fica ativo. Neste momento o endereço de memória já teve tempo de se estabilizar e assim a transição negativa de $\overline{MREQ}$ pode ser usada diretamente para habilitar as pastilhas de memória dinâmica. O sinal $\overline{RD}$ também fica ativo para indicar que os dados lidos da memória devem ser habilitados para dentro da barra de dados da CPU. A CPU colhe o dado da memória pela barra de dados durante a transição positiva de clock de estado T_3, e esta mesma transição é usada pela CPU para desativar os sinais $\overline{RD}$ e $\overline{MRQ}$. Portanto, o dado já foi colhido pela CPU antes que o sinal $\overline{RD}$ fique desativado. Os estados T_3 e T_4 de um ciclo de busca são utilizados para refrescar memórias dinâmicas. (A CPU usa este tempo para decodificar e executar a instrução colhida e então nenhuma outra operação poderia ser realizada nesta hora). Durante T_3 e T_4 os 7 bits menos significativos da barra de endereços contêm o endereço de restauração da memória e o sinal $\overline{RFSH}$ fica ativo para indicar que uma leitura de restauração de todas as memórias dinâmicas deve ser realizada. Observe que um sinal de $\overline{RD}$ não é gerado durante o tempo de restauração, para prevenir que os dados de diferentes segmentos de memória sejam colocados em conflito na barra de dados. O sinal $\overline{MREQ}$ durante a restauração deve ser usado para executar uma leitura de restauração; não pode ser utilizado até que o endereço de restauração esteja garantidamente estável pelo tempo de $\overline{MREQ}$.

Busca de OP CODE da Instrução
Figura 1

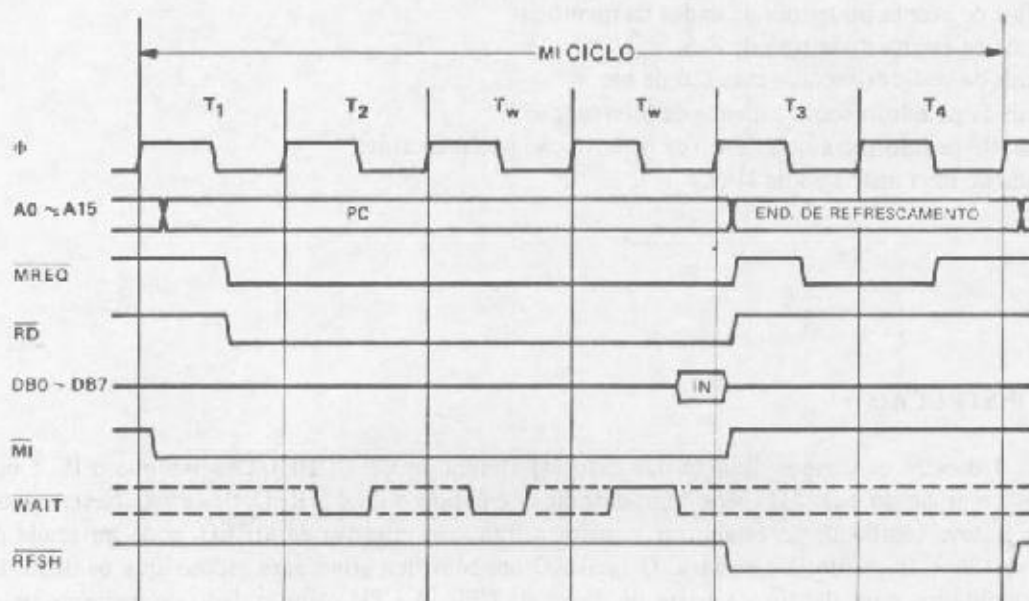
A figura 1A ilustra como um ciclo de busca seria retardado se a memória ativasse a linha de $\overline{WAIT}$. Durante o estado T_2 e de todos os T_W subseqüentes, a CPU amostra a linha de $\overline{WAIT}$ na transição negativa de 0. Se a linha de $\overline{WAIT}$ estiver ativa nesta hora, um outro estado de WAIT será introduzido no ciclo seguinte. Usando-se esta técnica, o ciclo de leitura pode ser estendido para adaptar-se ao tempo de acesso de qualquer tipo de memória.

Busca de OP CODE de Instrução com Ciclos de WAIT (Espera)

Figura 1A

ESCRITA OU LEITURA DA MEMÓRIA

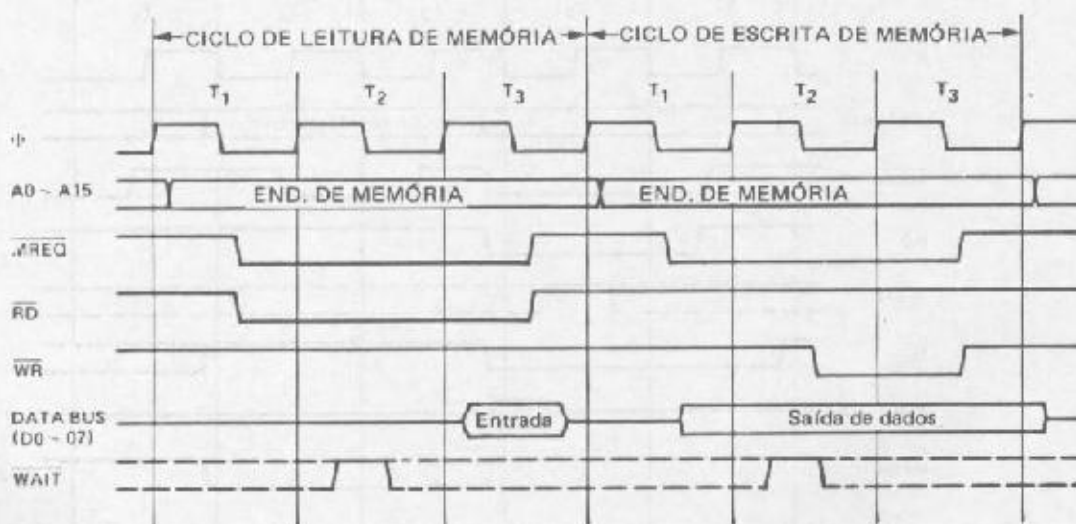
A figura 2 ilustra os tempos dos ciclos de escrita e leitura da memória, diferentes do ciclo M1 (busca do OP CODE). Estes ciclos são geralmente da duração de 3 períodos de clock, a menos que sejam solicitados pela memória, estados de WAIT, via sinal de $\overline{WAIT}$. Os sinais $\overline{RD}$ e $\overline{MREQ}$ são utilizados da mesma forma que no ciclo de busca. No caso de um ciclo de escrita, o $\overline{MREQ}$ fica ativo quando a barra de endereços estiver estável e então pode ser utilizado diretamente como "CHIP ENABLE" para memórias dinâmicas. O sinal $\overline{WR}$ fica ativo quando o dado na barra de dados estiver estável e então pode ser usado diretamente como um pulso R/W para quase qualquer tipo de memória semicondutora. O sinal WR fica inativo meio estado T antes do conteúdo das barras de dados e endereços serem trocados e então as necessidades de tempo de quase todos os tipos de memória semicondutora são satisfeitas.

Ciclos de escrita ou leitura de memória

Figura 2

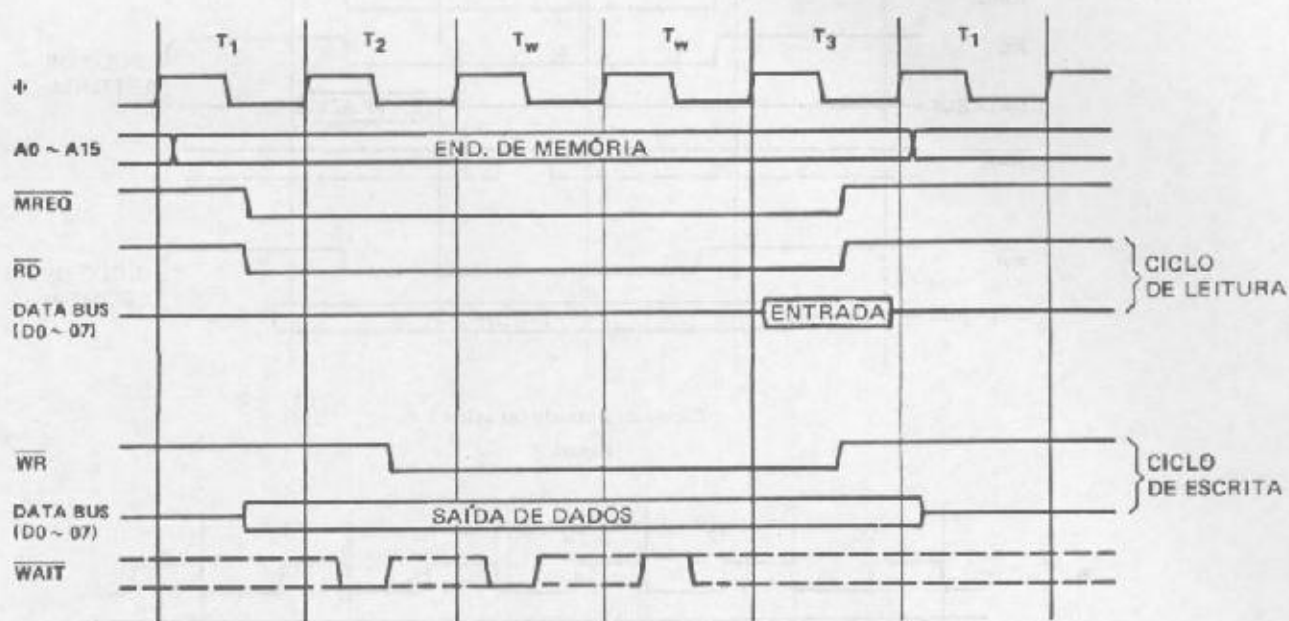
A figura 2A ilustra como um sinal de pedido de $\overline{WAIT}$ estenderá qualquer operação de leitura ou escrita na memória. Esta operação é idêntica àquela previamente descrita para o ciclo de busca. Observe nesta figura que um ciclo separado de leitura e escrita são mostrados na mesma figura embora estes ciclos nunca possam ocorrer simultaneamente.

Ciclos de escrita e leitura na memória com estados de espera (WAIT)

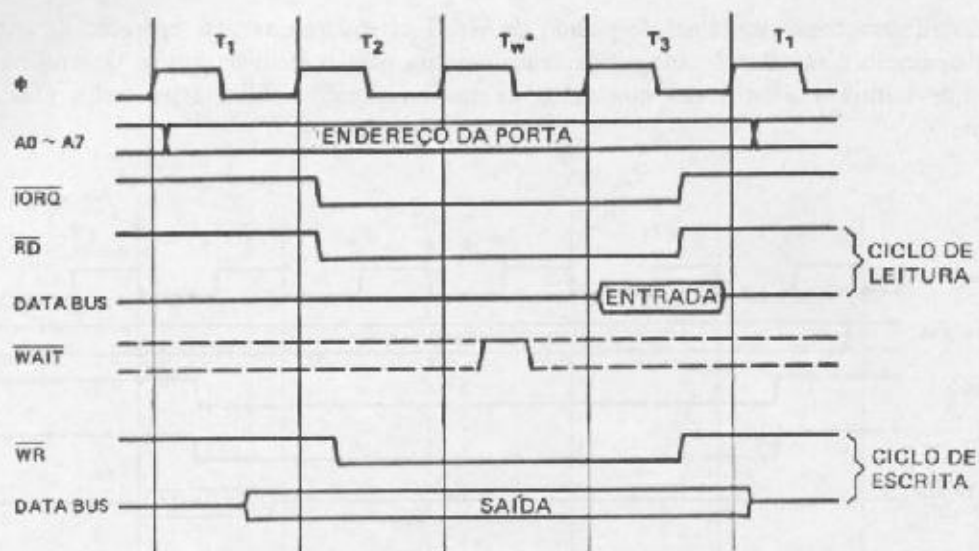
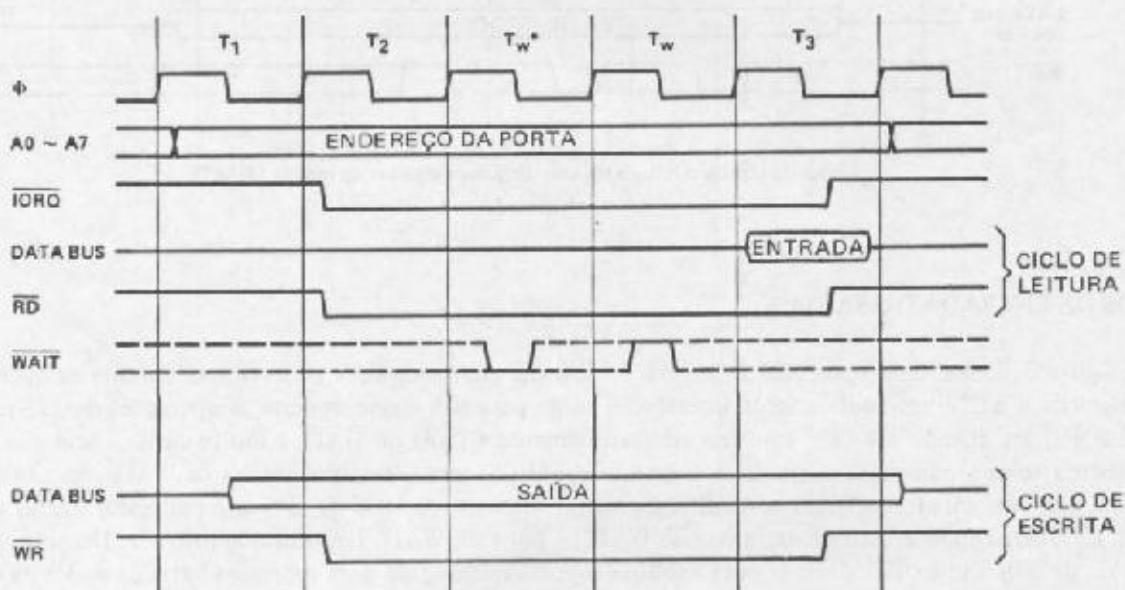
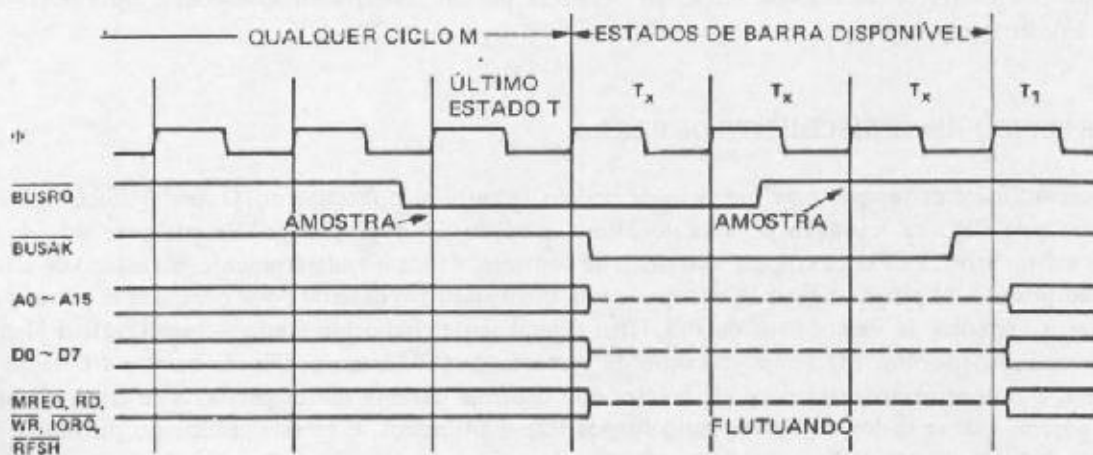
Figura 2A

CICLOS DE ENTRADA OU SAÍDA

A figura 3 ilustra uma operação de leitura de E/S ou escrita de E/S. Observe que durante as operações de E/S um estado de WAIT é automaticamente inserido. A razão para isto é que durante as operações de E/S o tempo, entre o sinal $\overline{IORQ}$ ser ativado e a CPU amostrar obrigatoriamente a linha de $\overline{WAIT}$, é muito curto, e sem este estado extra, não existiria tempo para uma porta de E/S decodificar seu endereço e ativar a linha de $\overline{WAIT}$ caso fosse necessário. Também, sem este estado de WAIT seria difícil projetar dispositivos MOS de E/S que pudessem operar na velocidade normal da CPU. Durante este estado extra de WAIT, a linha de $\overline{WAIT}$ também é amostrada. Durante uma operação de leitura de E/S a linha $\overline{RD}$ é usada para habilitar a porta endereçada para entrar na barra de dados da mesma forma que no caso de leitura de memória. Para operações de escrita de E/S, a linha $\overline{WR}$ é utilizada como clock para a porta endereçada, com tempo suficiente de superposição fornecido automaticamente de modo que a transição positiva possa ser usada como clock de dados.

A figura 3A ilustra como estados adicionais de WAIT podem ser acrescentados com a linha $\overline{WAIT}$. A operação é idêntica à descrita previamente.

CICLO DE PEDIDO/RECONHECIMENTO DE BARRA

A figura 4 ilustra os tempos para um ciclo de pedido de barra/reconhecimento. O sinal $\overline{BUSRQ}$ (pedido de barra) é amostrado pela CPU na transição positiva do último período de clock (estado) de qualquer ciclo de máquina. Se o $\overline{BUSRQ}$ estiver ativo, a CPU irá colocar seus sinais de controle, dados e endereçamento em estado de alta impedância na transição positiva do próximo clock. Neste momento, um dispositivo externo pode controlar as barras para transferir dados entre a memória de dispositivos de E/S. (Isto é geralmente conhecido como Acesso Direto à Memória [ADM] usando ciclos independentes.) O tempo máximo de resposta da CPU a um pedido de barra é a duração de um ciclo de máquina, e o controlador externo pode manter este controle durante tantos períodos de clock quanto desejados. Observe, porém, que se ciclos de ADM muito longos forem utilizados, e se está utilizando memórias dinâmicas, o controlador externo deverá realizar também a função de refrescamento. Esta situação só ocorre se blocos de dados muito grandes são transferidos sob o controle de ADM. Note também que durante um ciclo de pedido de barra a CPU não poderá ser interrompida nem por um sinal $\overline{INT}$ nem por um sinal $\overline{NMI}$.

Ciclos de entrada ou saída
Figura 3

Ciclos de entrada/saída com estados de espera
Figura 3A

Ciclo de pedido/reconhecimento de barra
Figura 4

CICLO DE PEDIDO/RECONHECIMENTO DE INTERRUPÇÃO

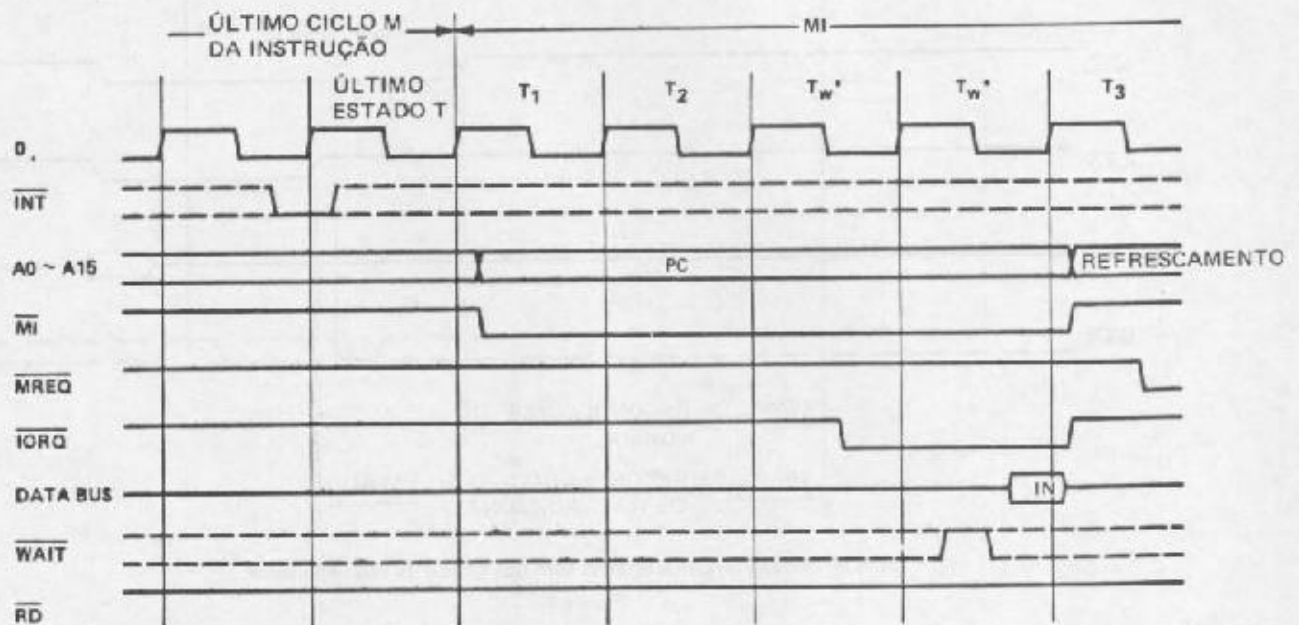
A figura 5 ilustra os tempos associados a um ciclo de interrupção. O sinal de interrupção ($\overline{INT}$) é amostrado pela CPU na transição positiva do último clock do final de qualquer instrução. O sinal não será aceito caso os flip-flops internos de habilitação de interrupção (controlados por software) não estiverem ativados ou se o sinal $\overline{BUSRQ}$ estiver ativo. Quando o sinal é aceito, um ciclo M1 especial é gerado. Durante este M1 especial, o sinal $\overline{IORQ}$ fica ativo (no lugar do $\overline{MREQ}$ normal) para indicar que o dispositivo que interrompeu pode colocar, na barra de dados, um vetor de 8 bits. Observe que 2 estados de WAIT são automaticamente inseridos neste ciclo. Estes estados são adicionados para que um esquema de prioridade de interrupção por transição possa ser facilmente implementado. Estes dois estados permitem tempo suficiente para os sinais de transição estabilizarem e identificarem qual o dispositivo de E/S que deverá inserir o vetor de resposta.

Ciclo de pedido de interrupção/reconhecimento

Figura 5

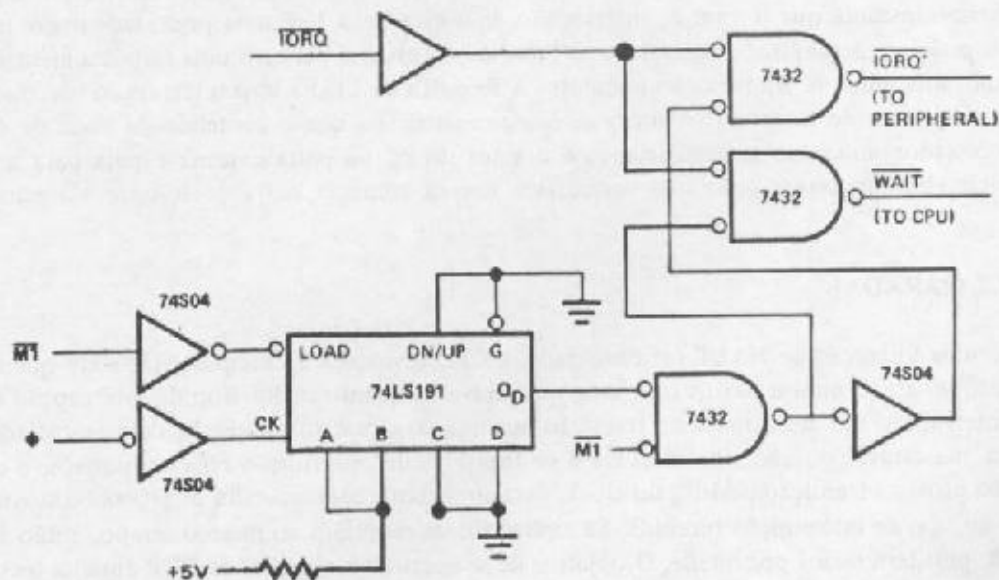
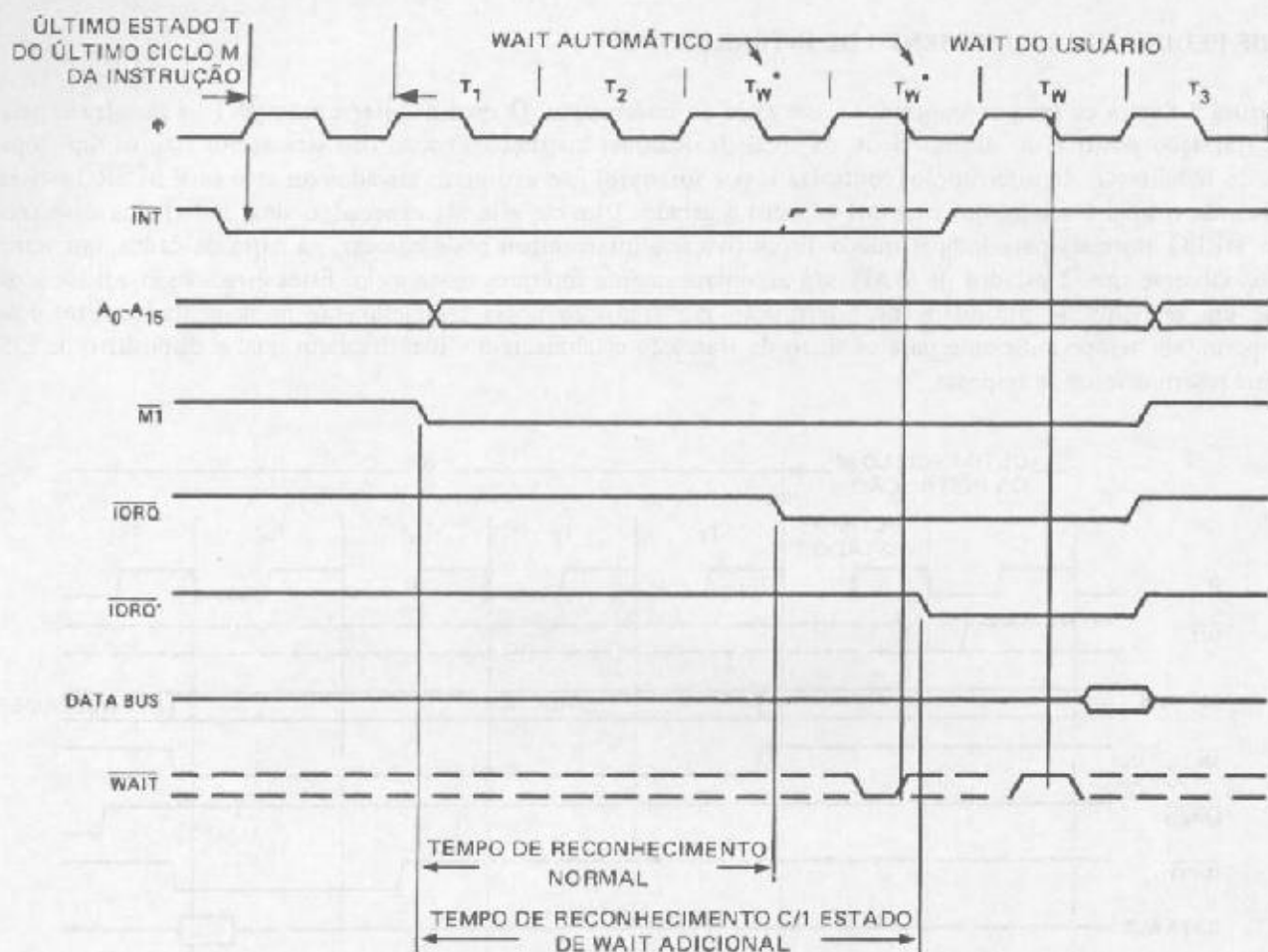
As figuras 5A e 5B mostram como um contador programável pode ser utilizado para estender o tempo de reconhecimento de interrupção. (Configurado para acrescentar 1 estado de 1 WAIT),

Tempo de reconhecimento de interrupção estendido com estado de espera (Wait)

Figura 5A

Ciclo de pedido/reconhecimento com um estado de Wait adicional
Figura 5B

RESPOSTA A INTERRUPÇÃO NÃO MASCARÁVEL

A figura 6 ilustra o ciclo de pedido/reconhecimento de uma interrupção não mascarável. Este sinal ($\overline{NMI}$) é amostrado no mesmo instante que o sinal de interrupção, mas esta linha tem uma prioridade maior que a interrupção normal e ela não pode ser desabilitada por software. Sua função usual é permitir uma resposta imediata a importantes eventos, tais como uma falha de alimentação iminente. A resposta da CPU a uma interrupção não mascarável é similar a uma operação de leitura de memória. A única diferença existente é que o conteúdo da barra de dados é ignorado enquanto o processador armazena automaticamente o valor de PC na pilha externa e pula para a posição 0066_H. A rotina de tratamento da interrupção não mascarável deverá começar nesta posição se ela estiver sendo usada.

SAÍDA DE HALT (PARADA)

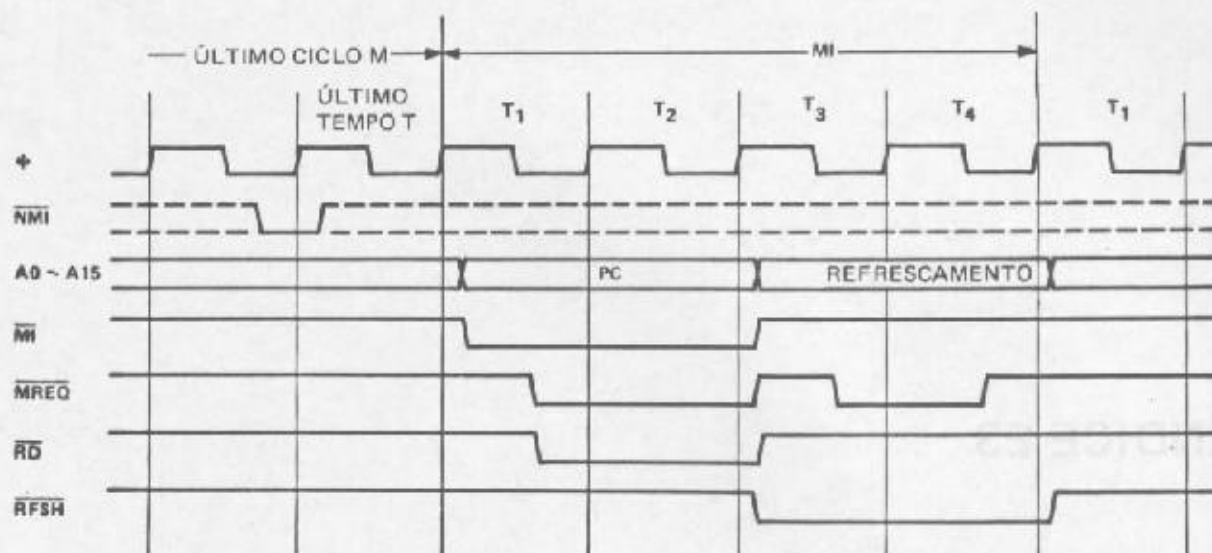
Sempre que uma instrução de HALT for executada, a CPU começará a executar NOP's até que uma interrupção seja enviada (tanto uma não mascarável quanto uma mascarável enquanto o flip-flop de interrupção estiver ativo). As duas linhas da interrupção são amostradas na transição positiva do clock do estado T_4 como mostrado na figura 7. Se uma interrupção, mascarável ou não, for recebida e os flip-flops de interrupção estiverem ativos, o estado de HALT será terminado na próxima transição positiva do clock. O ciclo seguinte será um ciclo de reconhecimento de interrupção correspondente ao tipo de interrupção recebida. Se ambas forem recebidas ao mesmo tempo, então a não mascarável será reconhecida, pois tem maior prioridade. O objetivo de se executar instruções de NOP durante o estado de HALT é manter os sinais de restauração da memória ativos. Cada ciclo no estado de HALT é ciclo M1 normal (busca) exceto que os dados recebidos da memória são ignorados e uma instrução de NOP é forçada internamente para a CPU. O sinal de recolhimento de HALT fica ativo durante este tempo para indicar que o processador está em estado de HALT.

Operação de pedido de interrupção não mascarável

Figura 6

Saída de HALT

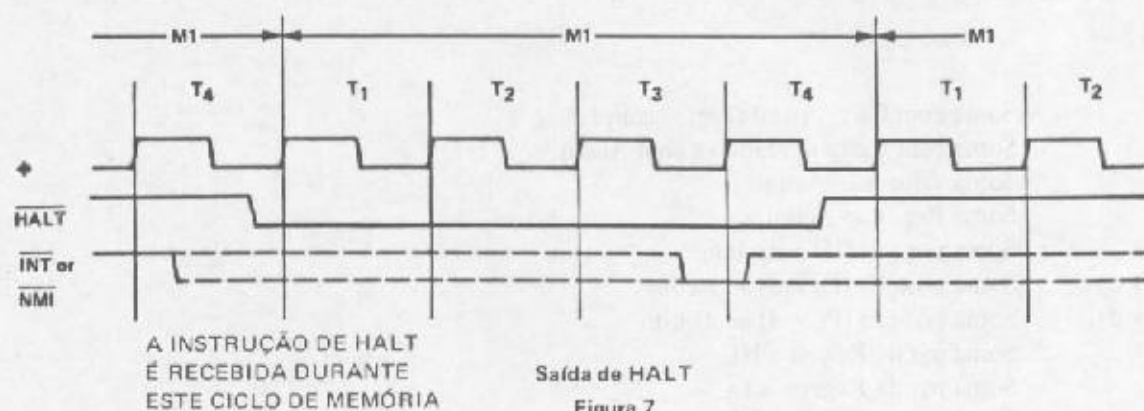



Figura 7

APÊNDICE E3

SUMÁRIO DO CONJUNTO DE INSTRUÇÕES

ADC HL, ss	Soma com Carry, par de reg. ss com HL.
ADC A, s	Soma com Carry operando s com Acum.
ADD A, n	Soma valor n ao Acum.
ADD A, r	Soma Reg. r ao Acum.
ADD A, (HL)	Soma posição (HL) ao Acum.
ADD A, (IX + d)	Soma posição (IX + d) ao Acum.
ADD A, (IY + d)	Soma posição (IY + d) ao Acum.
ADD HL, ss	Soma par de Reg. ss a HL.
ADD IX, pp	Soma par de Reg. pp a IX.
ADD IY, rr	Soma par de Reg. rr a IY.
AND s	"E" Lógico do operando s com Acum.
BIT b, (HL)	Testa bit b da posição (HL).
BIT b, (IX + d)	Testa bit b da posição (IX + d).
BIT b, (IY + d)	Testa bit b da posição (IY + d)
BIT b, r	Testa bit b do Reg. R.
CALL cc, nn	Chama sub-rotina na posição nn se a condição cc for verdade.
CALL nn	Chamada de sub-rotina incondicional na posição nn.
CCF	Complementa o Carry.
CP s	Compara operando s com Acum.
CPD	Compara posição (HL) com Acum, decrementa HL e BC e repete até que BC = 0.
CPI	Compara posição (HL) com Acum.
	Incrementa HL e decrementa BC.
CPIR	Compara posição (HL) e Aum.
	Incrementa HL, decrementa BC e repete até BC = 0.
CPL	Complementa Acum.
DAA	Ajuste decimal no Acum.
DEC m	Decrementa operando m.
DEC IX	Decrementa IX.
DEC IY	Decrementa IY.
DEC ss	Decrementa par de Reg. ss.
DI	Desabilita interrupções.

DJNZ e	Decrementa B e desvia relativo se B $\neq$ 0.
EI	Habilita interrupções.
EX (SP), HL	Troca a posição (SP) com HL.
EX (SP), IX	Troca a posição (SP) com IX.
EX (SP), IY	Troca a posição (SP) com IY.
EX AF, AF′	Troca o conteúdo de AF com AF .
EX DE, HL	Troca o conteúdo de DE com HL.
EXX	Troca o conteúdo de BC, DE e HL com os de BC′, DE′ e HL′ respectivamente.
HALT	PARADA (espera por int. ou reset).
IM 0	Configura modo de interrupção 0.
IM 1	Configura modo de interrupção 1.
IM 2	Configura modo de interrupção 2.
IN A, (n)	Carrega o ACUM com entrada do dispositivo (n).
IN R, (c)	Carrega Reg. r com entrada do dispositivo (c).
INC (HL)	Incrementa posição (HL).
INC IX	Incrementa IX.
INC (IX + d)	Incrementa posição (IX + d).
INC IY	Incrementa IY.
INC (IY + d)	Incrementa posição (IY + d).
INC r	Incrementa Reg. r.
INC ss	Incrementa par de reg. ss.
IND	Carrega posição (HL) com entrada da porta (c) e decrementa HL e B.
INDR	Carrega posição (HL) com entrada da porta (c), decrementa HL e B, repete até B = 0.
INI	Carrega posição (HL) com entrada da porta (c), incrementa HL e decrementa B.
INIR	Carrega posição (HL) com entrada da porta (c), incrementa HL, decrementa B e repete até B = 0.
JP (HL)	Desvio incondicional para (HL).
JP (IX)	Desvio incondicional para (IX).
JP (IY)	Desvio incondicional para (IY).
JP cc, nn	Desvio para posição nn se a condição cc for verdade.
JP nn	Desvio incondicional para posição nn.
JP C, e	Desvio relativo para PC + e se Carry = 1.
JR e	Desvio relativo incondicional para PC + e
JP NC, e	Desvio relativo para PC + e se Carry = 0.
JP NZ, e	Desvio relativo para PC + e se Z = 0.
JR Z, e	Desvio relativo para PC + e se Z = 1.
LD A, (BC)	Carrega Acum. com posição (BC).
LD A, (DE)	Carrega Acum. com posição (DE).
LD A, I	Carrega Acum. com Reg. I.
LD A, (nn)	Carrega Acum. com posição (nn).
LD A, R	Carrega Acum. com Reg. R.
LD (BC), A	Carrega posição (BC) com Acum.
LD (DE), A	Carrega posição (DE) com Acum.
LD (HL), n	Carrega posição (HL) com valor n.
LD dd, nn	Carrega par de Reg. dd com valor nn.
LD HL, (nn)	Carrega HL com posição (nn).
LD (HL), r	Carrega posição (HL) com Reg. r.
LD I, A	Carrega Reg. I com Acum.
LD IX, nn	Carrega IX com valor nn.
LD (IX + d), n	Carrega posição (IX + d) com valor n.
LD (IX + d), r	Carrega posição (IX + d) com Reg. r.
LD IY, nn	Carrega IY com valor nn.
LD IY, (nn)	Carrega IY com posição nn.
LD (IY + d), n	Carrega posição (IY + d) com valor n.
LD (IY + d), r	Carrega posição (IY + d) com Reg. r.
LD (nn), A	Carrega posição (nn) com Acum.
LD (nn), dd	Carrega posição (nn) com par Reg. dd.
LD (nn), HL	Carrega posição (nn) com HL.

LD (nn), IX	Carrega posição (nn) com IX.
LD (nn), IY	Carrega posição (nn) com IY.
LD R, A	Carrega Reg. R com Acum.
LD r, (HL)	Carrega Reg. r com posição (HL).
LD r, (IX + d)	Carrega Reg. r com posição (IX + d).
LD r, (IY + d)	Carrega Reg. r com posição (IY + d)
LD r, n	Carrega Reg. r com valor n.
LD r, r'	Carrega Reg. r com Reg. r'.
LD SP, HL	Carrega SP com HL.
LD SP, IX	Carrega SP com IX.
LD SP, IY	Carrega SP com IY.
LDD	Carrega posição (DE) com posição (HL), decrementa DE, HL e BC.
LDDR	Carrega posição (DE) com posição (HL), decrementa DE, HL e BC. Repete até BC = 0.
LDI	Carrega posição (DE) com posição (HL), incrementa DE, HL e decrementa BC.
LDIR	Carrega posição (DE) com posição (HL), incrementa DE, HL e decrementa BC. Repete até BC = 0.
NEG	Nega Acum. (complemento A2)
NOP	Não opera.
OR s	"OU" lógico entre operando s e Acum.
OTDR	Carrega porta (c) de saída com posição (HL), decrementa HL, decrementa B e repete até B =0.
OTIR	Carrega porta (c) de saída com a posição (HL), incrementa HL, decrementa B e repete até B = 0.
OUT(c), r	Carrega porta (c) de saída com Reg. r.
OUT(n), A	Carrega porta (n) de saída com Acum.
OUTD	Carrega porta (c) de saída com posição (HL), decrementa HL e B.
OUTI	Carrega porta (c) de saída com posição (HL), incrementa HL e decrementa B.
POP IX	Carrega IX com o topo de pilha.
POP IY	Carrega IY com o topo da pilha.
POP qq	Carrega par de Reg. com o topo da pilha.
PUSH IX	Carrega IX na pilha.
PUSH IY	Carrega IY na pilha.
PUSH qq	Carrega par de Reg. na pilha.
RES b, m	Limpa bit b do operando m.
RET	Retorno de sub-rotina.
RET cc	Retorno de sub-rotina se a condição cc for verdade.
RETI	Retorno de interrupção.
RETN	Retorno de interrupção não mascarável.
RL m	Giro à esquerda com carry do operando m.
RLA	Giro à esquerda do acumulador com carry.
RLC (HL)	Giro à esquerda circular da posição (HL).
RLC (IX + d)	Giro à esquerda circular da posição (IX + d).
RLC (IY + d)	Giro à esquerda circular da posição (IY + d).
RLC r	Giro à esquerda circular do Reg. r.
RLCA	Giro à esquerda circular do Acum.
RLD	Giro de dígito à esquerda e à direita entre Acum. e posição (HL).
RR m	Giro à direita do operando m com carry.
RRA	Giro à direita do Acum. com carry.
RRC m	Giro à direita circular do operando m.
RRC A.	Giro à direita circular do Acum.
RRD	Giro de dígito à direita e à esquerda entre Acum. e posição (HL).
RST p	Restart para posição p.
SBC A, s	Subtrai operando s do Acum. com carry.
SBC HL, ss	Subtrai par de Reg. ss de HL com carry.
SCF	Ativa o carry (C = 1).
SET b, (HL)	Ativa bit b de posição (HL).
SET b, (IX + d)	Ativa bit b da posição (IX + d).
SET b, (IY + d)	Ativa bit b da posição (IY + d).
SET b, r	Ativa bit b do Reg. r.
SLA m	Deslocamento aritmético à esquerda do operando m.
SRA m	Deslocamento aritmético à direita do operando m.

SRL m	Deslocamento lógico à direita do operando m.
SUB s	Subtrai operando s do Acum.
XOR s	"OU EXCLUSIVO" entre operando s e Acum.

GLOSSÁRIO

Acoplador acústico – Um dispositivo que permite um terminal ser conectado ao computador via uma linha telefônica. Ele é conectado ao aparelho telefônico.

Acumulador – Um registro temporário onde os resultados dos cálculos podem ser armazenados pelo processador central. Um ou mais acumuladores podem fazer parte da unidade lógica aritmética.

Algoritmo – Uma solução passo a passo para um problema, em um número de passos finito. Um procedimento específico para alcançar um resultado desejado.

Armazenamento de massa – Discos flexíveis, cassetes ou fitas usados para armazenar grandes quantidades de dados. Menos acessível, porém maior do que a memória principal.

Arquivo – Um conjunto de registros de gravação relacionados, tratados como uma unidade.

ASCII – "American Standard Code for Information Interchange". Código padrão de 7 bits amplamente utilizado. Também conhecido como USASCII; a IBM usa o EBCDIC, que tem 8 bits.

BASIC – "Beginner's All-purpose Symbolic Instruction Code". Linguagem algébrica desenvolvida no Dartmouth College. Essa linguagem é de simples aprendizado e utilização.

Binário – Um sistema de numeração baseado em múltiplos de dois, usando os dígitos 0 e 1.

Bit – Abreviação de dígito binário. Um único elemento em um número binário – tanto um 0 como um 1. Os bits são representados num microcomputador pelo estado do chaveamento eletrônico que pode ser "ligado" ou "desligado". Quatro bits formam um nibble e oito bits formam um byte.

Bit mais significativo – O dígito binário ocupando a posição mais à esquerda em um número ou palavra, normalmente 2^7 ou 128.

Bit menos significativo – O dígito binário ocupando a posição mais à direita em um número ou palavra, i.e: 2^0 ou 1.

Byte – Um grupo de bits adjacentes, normalmente oito bits, que é operado como sendo uma unidade pelo processador central.

Clock – Um dispositivo que gera pulsos reguladores e que sincroniza os eventos através de um microcomputador.

COMS – "Complementary Metal-Oxide Semiconductor". Uma tecnologia que combina a densidade de componente do MOS (PMOS) canal p e a velocidade de MOS (NMOS) canal n. O consumo de potência é muito baixo.

Compilador – Um programa que transforma uma linguagem de programação em alto nível em linguagem de máquina. Pode produzir várias microinstruções para cada instrução de alto nível, ao contrário do montador que transforma item para item. Quando se usa um compilador, não se pode alterar um programa sem recompilá-lo depois.

Digital – Pertinente a números inteiros discretos numa determinada base que pode expressar todas as variáveis que ocorrem num problema. Representado eletronicamente por 2 (binário) até 16 (hexadecimal) estados até o momento. Contrasta com o analógico, que se refere a um alcance contínuo de quantidades de tensão ou de corrente.

Dupla densidade – Método de dobrar a densidade de bits nos meios magnéticos de armazenamento.

EBCDIC – Código de 8 bits da IBM, similar ao ASCII.

Editor – Um programa que rearruma textos. Permite a adição ou deleção de símbolos e alterações de formato.

EIA – RS – 232 C – Interface padrão para transmissão serial de dados que não é síncrona com o processador central.

Endereço – Um número identificador ou em rótulo para posições na memória.

EPROM – "Erasable – Programmable Read-Only Memory". Uma PROM que pode ser apagada e reprogramada. Algumas EPROMs tem uma janela de quartzo em cima do chip, os dados podem ser apagados pela exposição à luz ultravioleta intensa; outras EPROMs podem ser apagadas eletricamente.

Flag (Bandeira) – Um bit vinculado a uma palavra para identificação ou para sinalização de alguma condução. Os microprocessadores típicos possuem flags para os estados de carry, zero, sinal, transbordo e semi-carry.

FSK – "Frequency Shift Keying". Técnica de transformar bits em duas freqüências diferentes representando 0 e 1 para transmissão em linhas telefônicas e de rádio. O dispositivo de interface é chamado de modem.

Hard-Copy – Saída impressa em papel.

Hardware – Os componentes físicos, periféricos, ou outros equipamentos que compõem um computador. Contrasta com software.

Hexadecimal – Um sistema de numeração baseado em múltiplos de 16 usando os caracteres de 0 a 9 e de A a F. Por exemplo, 0B em hexadecimal equivale a 00001011 em binário. Um byte pode ser codificado em exatamente 2 símbolos hexadecimais.

Instrução – Um passo de um programa que define uma operação juntamente com o(s) endereço(s) de qualquer dado necessário para esta operação.

Interface – Uma fronteira comum entre dois sistemas ou dispositivos. Ou hardware ou software necessários para interconectar duas partes de um sistema.

Interrupção – Uma paralisação na execução de um programa normalmente ocasionada por um sinal de um dispositivo externo.

Kansas City Standard – Refere-se a um padrão para gravações em fita cassete de dados EIA-RS-232 C. Oito ciclos de 2400 Hz formam um 1 e quatro ciclos de 1200 Hz formam um 0.

LIFO – "Last in-First Out". Método de acessos à entrada mais recente, e depois à próxima mais recente, e assim por diante. Em português: "Último a entrar, primeiro a sair".

Light pen – dispositivo fotossensível que pode ser usado para alterar a tela de um TRC gerando um pulso no ponto de contato.

Linguagem de alto nível – Uma linguagem de programação que é relativamente independente do montador e da linguagem de máquina. A gramática freqüentemente imita o Inglês e necessita de um compilador ou interpretador para convertê-la para código executável. Exemplos: BASIC, FORTRAN, COBOL, ALGOL, PL/M, APL.

Linguagem de máquina – Conjunto de inteiros binários que podem ser diretamente executados como instruções pelos microcomputadores sem interpretação prévia.

Memória – Dispositivo de armazenamento de informações binárias.

Memória dinâmica – Armazenamento de dados em chips dinâmicos, onde o armazenamento de uma pequena carga indica um bit. Devido à perda desta carga com o tempo, as memórias dinâmicas devem ser periodicamente restauradas.

Memória programável – Armazenamento no qual o acesso a novas informações é independente do endereço previamente examinado.

Memória só de leitura – Read-Only Memory (ROM) – Armazenamento que não pode ser alterado. A informação é escrita durante a fabricação.

Microcomputador – Um pequeno sistema de computador capaz de realizar um repertório básico de instruções. Inclui um processador central, freqüentemente contido num único chip, memória, dispositivos de E/S e fonte de alimentação.

Microprocessador – Um processador central num único chip. Um processador completo num chip, fabricado utilizando-se técnicas de fabricação de microminiaturas, conhecidas como LSI (Large Scale Integration – Integração em alta escala).

Modem – MOdulador-DEModulador. Dispositivo que transforma dados binários em freqüências apropriadas para transmissão através de linhas telefônicas.

Monitor – Um programa que controla a operação de rotinas básicas para otimizar o tempo do computador.

Montador – Um programa que converte instruções simbólicas em macroinstruções de máquina.

Octal – Um sistema de numeração baseada em múltiplos de oito usando os dígitos de 0 a 7. Atualmente bastante superado pelo sistema hexadecimal.

Pacote de ponto flutuante – Conjunto de rotina de software que permite a alguns microcomputadores realizarem aritmética de ponto flutuante sem a adição extra de hardware.

Painel traseiro – Uma placa equipada com conectores interconectados por barras nas quais os módulos que compõem um computador podem ser inseridos. Também conhecido como placa de interligação ou placa-mãe.

Palavra – Um conjunto de bits que ocupa uma posição de armazenamento e é tratada como uma unidade. Pode ter qualquer número de bits, mas usualmente tem 4, 8 ou 16 bits.

Paridade – Um bit extra que indica se uma palavra de computador possui um número par ou ímpar de ls. Usada para detectar erros.

Periférico – Qualquer parte do equipamento, normalmente um dispositivo de E/S, vinculado ao processador central.

Pilha – Uma técnica de apresentação de programa seqüencialmente. Uma pilha é uma estrutura LIFO controlada por instruções de PUSH e POP.

Processador central – O processador central controla a operação de um microcomputador. O processador central pode buscar e armazenar dados e instruções da memória.

Processador de palavra – Um editor de textos que permite ao usuário modificar o texto: formatos, livros, cartas e relatórios.

Registro – Um dispositivo de memória, acessível diretamente pelo processador central, usado para o armazenamento temporário de uma palavra de computador durante operações aritméticas, lógicas ou de entrada/saída.

S-100 – Uma barra de 100 pinos usada no popular sistema 8080/Z80.

Sistema de desenvolvimento – Um sistema de microcomputador que possui todo o equipamento relacionado para o desenvolvimento de hardware e de software.

Sistema Operacional – Software que opera os recursos de hardware de um microcomputador. O sistema operacional pode fazer escalonamento, depuração, controle de E/S, contabilização, compilação, designação de armazenamento e gerenciamento de dados.

Software – Programas que traduzem linguagens de alto nível em linguagem de máquina, tais como, compiladores, sistemas operacionais, montadores, geradores, rotinas de bibliotecas e editores.

Terra – Ponto de referência elétrica de um circuito.

Tiny Basic – Basic pequeno – Linguagem de programação BASIC reduzida a uma forma simples que permite aritmética com inteiros e algumas operações com cadeias. O Tiny Basic normalmente ocupa 4 kg ou menos bytes de memória.

TRC – Tubo de Raios Catódicos. Um tubo eletrônico a vácuo que pode ser usado como uma tela gráfica. Também é usado como referência a terminais que incorporam um TRC. Em inglês CRT.

Tri-State – (três-estados) – Capacidade de existir em três estados lógicos – 0 (baixo), 1 (alto) e um estado indefinido (alta-impedância), i.e, flutuando.

UART – Transmissor Receptor Assíncrono Universal. Um transmissor que converte série para paralelo e vice-versa.

ÍNDICE ANALÍTICO

CADASTRO PARA MALA DIRETA

Favor preencher todos os campos

NOME (não abreviar):

END. RESIDENCIAL:

CIDADE: UF: CEP: FONE:

EMPRESA: SEXO: F ( ) M ( ) NASCIMENTO: ____ / ____ (MÊS / ANO)

1. Cargo:
( ) 1. Digitador ( ) 2. Programador
( ) 3. Analista ( ) 4. Supervisor
( ) 5. Gerente ( ) 6. Diretor
( ) 7. Outros (especificar):

2. Escolaridade: ( ) 1. 1º grau ( ) 2. 2º grau
( ) 3. 3º grau

3. Eu geralmente, compro livros:
( ) 1. Em livrarias ( ) 2. Por telefone
( ) 3. Em feiras ( ) 4. Por Reembolso Postal
( ) 5. Outros (especificar):

4. Costumo comprar em média, a seguinte quantidade de livros por ano:
( ) 1. 01 ( ) 2. 03 ( ) 3. 05 ( ) 4. mais

5. Estou interessado em livros das seguintes áreas:
( ) 1. Informática ( ) 2. Negócios ( ) 3. Controle Qualidade
( ) 4. Informática na Educação ( ) 5. Ensino Inglês ( ) 6. Psicologia
( ) 7. Sociologia ( ) 8. Metodologia ( ) 9. Medicina
( ) 10. Eng. Civil ( ) 11. Eng. Elétrica/Eletrônica ( ) 12. Eng. Mecânica
( ) 13. Eng. Química ( ) 14. Física ( ) 15. Matemática
( ) 16. Outras (especificar):

6. Utiliza microcomputador:
( ) Sim ( ) Não

7. Qual?
( ) 1. PC ( ) 2. Apple ( ) 3. MSX
( ) 4. Outros (especificar):

8. Softwares mais utilizados:
( ) 1. Banco de Dados ( ) 2. Planilha ( ) 3. Proc. Texto
( ) 4. Outros (especificar):

9. Linguagens:
( ) 1. C ( ) 2. Cobol
( ) 3. Clipper/dBase ( ) 4. Assembler
( ) 5. Outras (especificar):

Obra: Construa O Seu Próprio Microcomputador Z 80

Autoria: Steve Ciarcia

MAKRON Books

MAKRON *Books* do Brasil Editora Ltda.
Nova denominação da
Editora McGRAW-HILL Ltda.
"Padrão em Livros de Qualidade"
Rua Tabapuã, 1105 – Itaim Bibi – SP
04533 – Tels.: 820.6622/820.8528

Outros Livros na Área: